Chasing Bulges or Rotations? A Metamorphosis of the QR-Algorithm

Raf Vandebril

Raf Vandebril
Department of Computer Science
KU Leuven, Belgium
Raf.Vandebril@cs.kuleuven.be

Abstract

The QR-algorithm is a renowned method for computing all eigenvalues of an arbitrary matrix. A preliminary unitary similarity transformation to Hessenberg form is indispensable for keeping the computational complexity of the subsequent QR-steps under control. When restraining computing time is the vital issue, we observe that the prominent role played by the Hessenberg matrix is sufficient but perhaps not necessary to fulfill this goal. In this paper, a whole new family of matrices, sharing the major qualities of Hessenberg matrices, will be put forward. This gives rise to the development of innovative implicit QR-type algorithms, pursuing rotations instead of bulges. The key idea is to benefit from the QR-factorization of the matrices involved. The prescribed order of rotations in the decomposition of the Q-factor uniquely characterizes several matrix types such as Hessenberg, inverse Hessenberg, and CMV matrices. Loosening the fixed ordering of these rotations provides us the class of matrices under consideration. Establishing a new implicit QR-type algorithm for these matrices requires a generalization of diverse well-established concepts. We consider the preliminary unitary similarity transformation, a proof of uniqueness of this reduction, an extension of the CMV -decomposition to a double Hessenberg factorization, and an explicit and implicit QR-type algorithm. A detailed complexity analysis illustrates the competitiveness of the novel method with the traditional Hessenberg approach. The numerical experiments show comparable accuracy for a wide variety of matrix types, but disclose an intriguing difference between the average number of iterations before deflation can be applied.

Article information

- Vandebril, Raf. *Chasing bulges or rotations? A metamorphosis of the QR-algorithm*, SIAM Journal on Matrix Analysis and Applications, volume 32, issue 1, pages 217-247, 2011.
- The content of this article is identical to the content of the published paper, but without the final typesetting by the publisher.
- Journal's homepage: https://www.siam.org/journals/simax.php
- Published version: http://dx.doi.org/10.1137/100809167
- KU Leuven's repository url: https://lirias.kuleuven.be/handle/123456789/280648

SIAM J. MATRIX ANAL. APPL.
Vol. 32, No. 1, pp. 217–247

CHASING BULGES OR ROTATIONS? A METAMORPHOSIS OF THE QR-ALGORITHM*

RAF VANDEBRIL†

Abstract. The QR-algorithm is a renowned method for computing all eigenvalues of an arbitrary matrix. A preliminary unitary similarity transformation to Hessenberg form is indispensable for keeping the computational complexity of the subsequent QR-steps under control. When restraining computing time is the vital issue, we observe that the prominent role played by the Hessenberg matrix is sufficient but perhaps not necessary to fulfill this goal. In this paper, a whole new family of matrices, sharing the major qualities of Hessenberg matrices, will be put forward. This gives rise to the development of innovative implicit QR-type algorithms, pursuing rotations instead of bulges. The key idea is to benefit from the QR-factorization of the matrices involved. The prescribed order of rotations in the decomposition of the Q-factor uniquely characterizes several matrix types such as Hessenberg, inverse Hessenberg, and CMV matrices. Loosening the fixed ordering of these rotations provides us the class of matrices under consideration. Establishing a new implicit QR-type algorithm for these matrices requires a generalization of diverse well-established concepts. We consider the preliminary unitary similarity transformation, a proof of uniqueness of this reduction, an extension of the CMV-decomposition to a double Hessenberg factorization, and an explicit and implicit QR-type algorithm. A detailed complexity analysis illustrates the competitiveness of the novel method with the traditional Hessenberg approach. The numerical experiments show comparable accuracy for a wide variety of matrix types, but disclose an intriguing difference between the average number of iterations before deflation can be applied.

Key words. unitary similarity transformations, QR-type algorithms, Givens rotations, patterns of rotations, eigenvalues, (lower) Hessenberg, CMV-matrix

AMS subject classifications. 15A18, 15A21, 15A23

DOI. 10.1137/100809167

1. Introduction.

The prominent role of the Hessenberg matrix in nonsymmetric eigenvalue computations has—to some extent—always bothered me. Restraining computing time appears to be the driving force behind the ongoing investigations related to the Hessenberg structure and eigenvalue computations. We will see, however, that the Hessenberg matrix is just a member of a huge family of condensed matrix forms admitting a unified treatment of QR-type algorithms.

Among the so-called direct methods for eigenvalue computations, the QR-method is the most popular and most used one for determining the eigenvalues of nonsymmetric matrices. An extensive list of publications is devoted to this favorable technique. Introductions appear in almost all numerical linear algebra textbooks [13, 17, 21] and also more sophisticated books are available [19, 36, 41]. Detailed studies of the convergence behavior linking the QR-algorithm with subspace iteration can be found in [32, 37, 39]. There are articles on particular characteristics such as balancing, bulge chasing, and maintaining well-focused shifts [34, 38, 40] and even very recently several improvements to achieve speed-up and maintain high accuracy were proposed [6, 7].

*Received by the editors September 20, 2010; accepted for publication (in revised form) by M. E. Hochstenbach January 12, 2011; published electronically March 17, 2011. This work was supported by the "Postdoctoraal Onderzoeker" from the Fund for Scientific Research–Flanders (Belgium). The research was also partially supported by the Interuniversity Attraction Poles Programme, initiated by the Belgian State, Science Policy Office, Belgian Network DYSCO (Dynamical Systems, Control, and Optimization).

http://www.siam.org/journals/simax/32-1/80916.html

†Department of Computer Science, K.U.Leuven, 3001 Leuven (Heverlee), Belgium (raf.vandebril@cs.kuleuven.be, http://people.cs.kuleuven.be/raf.vandebril/).

Also a wide variety of publications, both theoretical [11, 12] and more practical, is related to studies on how to adapt the QR-algorithm to make it suitable for particular matrix structures such as quasi-separable [15], semiseparable (plus diagonal) [26, 28], unitary plus low rank [5, 23], Hermitian plus low rank [25], companion, comrade, Hamiltonian [16, 35], and many other matrices.

Important, however, for applying a QR-algorithm is to perform a preprocessing step transforming the involved matrix to a more economical format. This format is assumed to have some prominent features: it admits a cheaper QR-step, generically less memory is needed to store the matrix, and important is the preservation of the economical format after a QR-step. Well-known convenient formats are the tridiagonal form (for a Hermitian matrix) and the Hessenberg matrix (for a nonsymmetric matrix). The original matrix is brought, via unitary similarity transformations, to this shape retaining thereby the eigenvalues [13, 17, 36]. Some other less widespread formats are related to rank structured matrices [14, 22]. Also reduction algorithms tuned for specific matrix cases, for example, unitary, Hamiltonian, and symplectic, exist or are under investigation [3, 18].

In this article both the unitary similarity transformation and the QR-algorithm are reconsidered, generalizing thereby the results for Hessenberg and Hessenberg-like (inverses of Hessenberg) matrices. Examining the Q-factor in the QR-factorization of Hessenberg(-like) matrices, we observe that it can be factored by $n-1$ Givens rotations. The resulting unitary matrix Q is referred to as being in compressed format. For a Hessenberg matrix this compressed format is built by $n-1$ rotations in descending order, whereas a Hessenberg-like matrix has $n-1$ rotations in ascending order. Studies [1, 4, 8, 9, 20, 33] relating unitary matrices with orthogonal Laurent polynomials on the unit circle reveal, however, the existence of a wide variety of possible compressed factorizations of a unitary matrix. Instead of only considering ascending or descending sequences of rotations, an arbitrary number of changes in direction can be taken. For example, the nowadays popular CMV-decomposition presents a compressed factorization of a unitary matrix, altering the direction after each of the $n-1$ rotations.

Based on a prespecified pattern the compressed unitary matrix should satisfy, a new algorithm for executing the similarity transformation is presented resulting in a QR-factorization $\hat{A} = QR$, having Q in desired format. Overall, this method has the same computational complexity as the transformation to Hessenberg form. Both the Hessenberg and Hessenberg-like cases are contained as instances of this generic reduction method. Another example has the resulting matrix QR-factored, with Q decomposed in $n-1$ rotations according to the CMV-pattern. A proof of uniqueness of the reduction procedure based on Krylov subspaces is included [36]. Furthermore, a generalization of the CMV-decomposition is devised, admitting a factorization in two matrices having diagonal blocks of irreducible Hessenberg form of sizes exceeding 2×2. This decomposition gives rise to a product of a unitary upper and unitary lower Hessenberg matrix.

A QR-type algorithm is developed capable of dealing with the almost unlimited number of patterns that a matrix can be reduced to. Both an explicit as well as an implicit QR-type version suitable for the new structures are presented. Moreover, it will be possible to alter the patterns during execution. We will show that these new chasing algorithms share the computational complexity of their well-known instances: the Hessenberg and Hessenberg-like case. A compelling characteristic of this approach is that we will cease in chasing bulges but progress to a rotation chasing method.

Numerical experiments are conducted to test several variants of these new procedures. The accuracy for the presented approaches is comparable, but the execution time differs significantly. It will be shown that the average number of iterations is heavily dependent on the eigenvalue distribution.

The article is organized as follows. Section 2 presents preliminary results, definitions, and assumptions required for understanding the article. Section 3 discusses the algorithm for carrying out the similarity transformation to a prescribed matrix structure. In section 4, a factorization of a compressed unitary matrix in two Hessenberg matrices, extending the well-known CMV-decomposition, is presented. Uniqueness of the unitary similarity transformation is considered in section 5. In section 6 the new QR-type algorithm is presented as an explicit calculation and some issues such as the structure preservation and irreducibility are considered. In section 7 an implicit version of the generalized QR-method is derived. Some comments on the software, numerical issues, and tests are given in section 8. Conclusions and future work close the article.

2. Preliminary results. To obtain a self-contained article some definitions and techniques for factoring unitary matrices in rotations will be reviewed.

A *Hessenberg* matrix H has all elements below the subdiagonal zero. A *Hessenberg-like* matrix Z has all submatrices taken out of the lower triangular part of rank at most one: $\operatorname{rank} Z(i:n, 1:i) \leq 1$ for all $i = 1, \ldots, n$. In case of invertibility, the inverse of a Hessenberg matrix is of Hessenberg-like form. We tacitly assume that the reader is familiar with a Givens rotation and its ability of creating zeros in prescribed matrix positions by altering only two rows or columns of the involved matrix [17].

Remark 2.1. Throughout the manuscript we will continually use 2×2 rotations, though all results remain valid when using 2×2 unitary matrices instead. A rotation has determinant equal to 1, whereas the determinant of a unitary matrix can reach an arbitrary value of modulus equal to 1.

Rotations will be the building blocks for dealing with factorizations of unitary matrices. To be able to benefit from rotational factorizations it is essential that an elegant, intuitive, and compact manner is provided for keeping track of all information accompanying an individual rotation. A graphical depiction presents us readily the order of the rotations and indicates unambiguously the rows affected by the rotation.

This graphical representation is introduced by computing the QR-factorization of a matrix $A = (a_{ij})_{ij} \in \mathbb{C}^{5\times 5}$. The matrix elements are depicted by the symbol $\times$; each rotation is represented by a square bracket with arrows. The arrows point towards the rows affected when applying the rotation to the matrix. To compute the QR-factorization by means of rotations, first the lower left element a_{51} is annihilated by G_{51}^H. The matrix $G_{51}^H A$ has therefore a zero element in position $(5, 1)$. Graphically we obtain

$$G_{51}^H A = \begin{matrix} \\ \\ \\ \\ \\ \curvearrowright \end{matrix} \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \end{bmatrix} = \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \end{bmatrix}.$$

The procedure is continued by eliminating all but the top element of the first column. This results in $G_{21}^H G_{31}^H G_{41}^H G_{51}^H A = Q_1^H A$, graphically depicted as

$$Q_1^H A = \quad \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \end{bmatrix} \quad = \quad \begin{bmatrix} \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \end{bmatrix}.$$

In turn all columns are brought to upper triangular form by a succession of rotations $Q_4^H Q_3^H Q_2^H Q_1^H A = Q^H A = R$.

$$Q^H A = \quad \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \end{bmatrix} \quad = \quad \begin{bmatrix} \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times \\ 0 & 0 & 0 & \times & \times \\ 0 & 0 & 0 & 0 & \times \end{bmatrix}. \tag{2.1}$$

Premultiplication by Q leads to the QR-factorization of A, graphically represented in condensed format as (thereby not depicting zero elements anymore)

$$A = \quad \begin{bmatrix} \times & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & & \times & \times & \times \\ & & & \times & \times \\ & & & & \times \end{bmatrix}. \tag{2.2}$$

This schematic representation contains all vital information for operating on the QR-factorization of A. The brackets denote clearly on which rows the rotations act, and also the mutual order of the rotations is available. Moreover, one can read from these schemes which rotations (for example the two rotations in the third, fourth, or fifth column of the factorization of Q) commute and can be applied on R simultaneously or in either order. With a *rotational factorization* of Q we will refer to a factorization of Q in rotations.

Remark 2.2. We note that possibly in (2.2) some rotations need not be performed, since they equal the identity matrix. In the graphical schemes of rotational factorizations, rotations equal to the identity are, however, admitted. Only, when based on preliminary knowledge, such as the matrix structure, rotations equal to the identity will not be depicted in the schemes and as such reflect this particular structure.

Exploiting the matrix structure, a Hessenberg matrix H and a Hessenberg-like matrix Z admit QR-factorizations of the form (for 5×5 examples)

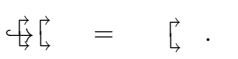
$$H = \quad \begin{bmatrix} \times & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & & \times & \times & \times \\ & & & \times & \times \\ & & & & \times \end{bmatrix} \quad \text{and} \quad Z = \quad \begin{bmatrix} \times & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & & \times & \times & \times \\ & & & \times & \times \\ & & & & \times \end{bmatrix}. \tag{2.3}$$

The sequence of rotations in the factorization of H is referred to as a *descending* sequence of rotations. The Hessenberg-like matrix on the contrary admits a rotational factorization of the involved unitary matrix consisting of an *ascending* sequence of rotations. One can easily verify that an ascending sequence of rotations has a one to one relation with a unitary upper Hessenberg matrix, whereas a descending sequence of rotations agrees with a unitary lower Hessenberg matrix. As will be shown in section 2.1, a rotational factorization is not always unique. When referring, however, to a very specific ordering of the rotations, as for a descending or ascending sequence, we will speak about the *shape* or the *pattern* of the rotations.

2.1. Manipulations with rotations. The algorithms presented in this paper depend heavily on manipulating rotations. These operations are already described and proved elsewhere (see, for example, [27, 28]). For completeness, however, we will briefly reconsider the necessary ones in this section, without proofs.

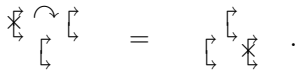
LEMMA 2.3 (fusion of rotations). *Suppose two* 2×2 *rotations* G_1 *and* G_2 *are given. Their product* $G_1 G_2 = G_3$ *is again a rotation. This operation is named the fusion of rotations.*

Graphically we depict this operation by a curly arrow as



LEMMA 2.4 (shift-through operation). *Suppose three* 3×3 *rotations* $\check{G}_1, \check{G}_2$, *and* $\check{G}_3$ *are given, such that the rotations* $\check{G}_1$ *and* $\check{G}_3$ *act on the first two rows of a matrix, and* $\check{G}_2$ *acts on the second and third row (when applied on the left to a matrix). Then there exist three rotations* $\hat{G}_1, \hat{G}_2$, *and* $\hat{G}_3$ *satisfying the equality* $\check{G}_1 \check{G}_2 \check{G}_3 = \hat{G}_1 \hat{G}_2 \hat{G}_3$, *where* $\hat{G}_1$ *and* $\hat{G}_3$ *work on the second and third row and* $\hat{G}_2$ *operates on the first two rows.*

This is a common result, whose proof is based on two variants for factoring a 3×3 unitary matrix [29]. Schematically this operation is depicted with arrows such as $\curvearrowright$ indicating the position were the marked rotation will move to

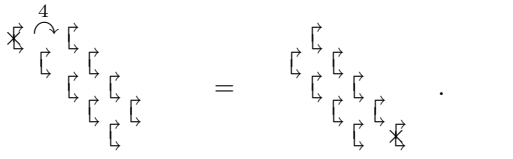


A shift-through operation of length ℓ is the final important operation.

LEMMA 2.5 (shift-through operation of length ℓ). *Suppose the following matrix product* $\check{G}\check{W}\check{X}$ *is given, where* $\check{G}$ *denotes a rotation acting on rows* 1 *and* 2. *The matrices* $\check{W}$ *and* $\check{X}$ *are unitary matrices, both having a rotational factorization in a descending sequence of* ℓ *rotations. The* i*th rotation* $\check{G}_i^W$ *of* $\check{W}$ *acts on rows* $i+1$ *and* $i+2$ *and the* i*th rotation* $\check{G}_i^X$ *related to* $\check{X}$ *acts on rows* i *and* $i+1$. *Then the matrix product* $\check{G}\check{W}\check{X}$ *can be refactored as* $\check{G}\check{W}\check{X} = \hat{W}\hat{X}\hat{G}$, *where* $\hat{G}$ *is a rotation acting on rows* $\ell+1$ *and* $\ell+2$. *The unitary matrices* $\hat{W}$ *and* $\hat{X}$ *are again descending sequences of* ℓ *rotations sharing the patterns of* $\check{W}$ *and* $\check{X}$, *meaning that the* i*th rotation* $\hat{G}_i^W$ *of* $\hat{W}$ *acts on rows* $i+1$ *and* $i+2$ *and the* i*th rotation* $\hat{G}_i^X$ *of* $\hat{X}$ *acts on rows* i *and* $i+1$

In scheme (2.4) the rotations $\check{G}$ and $\hat{G}$ are marked with a $\times$. On the left, the matrix product $\check{G}\check{W}\check{X}$ is shown, with $\check{W}$ the lower descending sequence of rotations and $\check{X}$ the upper sequence of rotations. A shift-through operation of a specified length is depicted by adding a superscript to an arrow $\curvearrowright$, standing for the number of successive shift-through operations that need to be performed. The arrow points to the final position of the marked rotation.

(2.4)



The right term in (2.4) admits now a rotational factorization $\hat{W}\hat{X}\hat{G}$ with $\hat{G}$ appearing in the bottom right; again $\hat{W}$ is the bottom and $\hat{X}$ the upper descending sequence of rotations.

In this article we will not limit ourselves to unitary matrices. For handling arbitrary matrices we will utilize their QR-factorization; this requires us to characterize also the interaction between rotations and upper triangular matrices R. For simplicity, we will confine ourselves to nonsingular matrices.[1] The next lemma specifies how sequences of rotations change when transferring them through an upper triangular matrix.

LEMMA 2.6. *A nonsingular $n \times n$ matrix A with RQ-factorization $A = \hat{R}\hat{Q}$ admits a QR-factorization $A = QR$, where the rotational factorizations of Q and $\hat{Q}$ obey the same pattern.*

A proof of this Lemma is available in [24]. In practice, transferring rotations through an upper triangular matrix affects the rotations themselves, but not the mutual positions nor the rows/columns they act on.

3. Unitary similarity transformations. A prescribed rotational pattern governs the flow of the similarity transformation. Operating on the QR-factorization of a matrix, the result will be a new QR-factorization, with the unitary factor in the desired shape. In the remainder of the text when speaking about a *similarity transformation*, a unitary similarity transformation is meant.

3.1. Admissible outcomes and compressed unitary matrices. Some particulars and extra notation are needed to specify without ambiguity the patterns of rotations of the Q-factor admitted as outcomes of the similarity transformation.

It is well-known that every unitary matrix is similar to a unitary Hessenberg matrix, admitting a Schur parameterization [8]. The Schur parameterization is a multiplicative factorization into a descending sequence of $n-1$ rotations. The coming definition of a compressed unitary matrix is a sort of extension of this factorization.

A vector $\mathbf{p}$ of length $n-2$ containing the characters ℓ (left) and r (right) will define the strict order of the rotations. Suppose $n-1$ rotations $G_i, i = 1, \ldots, n-1$ are given where G_i operates on rows i and $i+1$. The ith element $\mathbf{p}_i$ of the *position vector* $\mathbf{p}$ specifies the relative position of the factor G_i w.r.t. the factor G_{i+1} in the product of these rotations.

For example, the Q-factor in the QR-factorization of a Hessenberg matrix H can be factored in a descending sequence of rotations. This means that G_i always appears on the left of G_{i+1}; hence $Q = G_1 G_2 \ldots G_{n-1}$ and the corresponding $\mathbf{p} = [\ell, \ldots, \ell]$. The matrix Q itself is a unitary upper Hessenberg matrix. A Hessenberg-like matrix, however, is related to an ascending sequence of iterations; therefore $Q = G_{n-1} \ldots G_2 G_1$ and $\mathbf{p} = [r, \ldots, r]$. The matrix Q is now a unitary Hessenberg-like matrix or, because of the unitarity, a *unitary lower Hessenberg*; this is also how we will address these matrices in the rest of the article.

The CMV-decomposition [9, 20] of a unitary pentadiagonal matrix $P = UV$ factors the matrix in two block diagonal matrices U and V, where the blocks are 2×2 rotations

$$P = UV = \begin{bmatrix} \curvearrowright & & & & \\ & \curvearrowright & & & \\ & & \curvearrowright & & \\ & & & \curvearrowright & \\ & & & & \ddots \end{bmatrix} \begin{bmatrix} 1 & & & & \\ & \curvearrowright & & & \\ & & \curvearrowright & & \\ & & & \curvearrowright & \\ & & & & \ddots \end{bmatrix}. \tag{3.1}$$

[1] We note that Lemma 2.6 remains valid also in the singular case [24]. But, to exclude some technical and troublesome cases in the next parts on unitary similarity transformations and the QR-type method, this restriction is posed in this article.

Corresponding to our notation[2] $P = G_5G_3G_1G_2G_4$, where $U = G_5G_3G_1$ and $V = G_2G_4$. The position vector $\mathbf{p}$ equals $[\ell, r, \ell, r]$.

In the rest of the article we refer to a *compressed unitary* matrix as a unitary matrix admitting a factorization of $n-1$ rotations G_i acting on rows i and $i+1$, according to a prescribed ordering $\mathbf{p}$. The similarity transformation proposed in the next section admits all types of compressed unitary matrices $\hat{Q}$ in the QR-factorization of the resulting matrix $\hat{A} = \hat{Q}\hat{R}$, covering thereby the Hessenberg, lower Hessenberg, and the CMV case as particular examples. When speaking of a *compressed QR-factorization*, the word compressed alludes to the unitary factor. A top to bottom transversal of a pattern of rotations from the factorization of a compressed unitary matrix often involves a change of direction; hence we will refer to any of these patterns as *zigzag patterns*.

Factorizations of a unitary matrix in rotations were discussed before in several articles [1, 2, 4, 8, 9, 33]. The origin lies in the so-called Schur parameterization [8]. Already in these manuscripts results related to other types of patterns [1] were proposed. The relations between orthogonal Laurent polynomials and the corresponding unitary matrices were studied in [4, 9, 33]. Depending on the order of the monomials for constructing an orthogonal basis of Laurent polynomials, the recurrences are given by a compressed unitary matrix. Both the order of the monomials as well as the shape of the corresponding compressed unitary matrix are determined by a position vector $\mathbf{p}$ (see also section 5). In [4], these factorizations are referred to as snake-shaped matrix factorizations owing to the zigzag form of the pattern of rotations.

Remark 3.1. Though not explicitly mentioned, it is tacitly assumed that all unitary matrices have determinant equal to 1. Otherwise, an extra unimodular factor appears when building the factorization of a compressed unitary matrix. This unimodular factor does not pose any difficulties; all deductions remain valid just as the shift-through operations and the fusions, only all theorems need the incorporation of an additional unimodular factor. Instead of considering rotations one can also use 2×2 unitary matrices solving the problem immediately.

3.2. The algorithm. One can consider the pyramid shaped factorization of a generic unitary matrix Q in (2.2) as made up by four descending sequences (2.1), or as made up by four ascending sequences of rotations. This two-folded interpretation enables us to reduce any matrix to the desired compressed format specified by $\mathbf{p}$. First the algorithm is presented, followed by a justification in sections 3.2.1–3.2.3.

In each step i (for $i = 1, \ldots, n-2$) of the algorithm the following similarity transformation is executed:[3]

- (if $\mathbf{p}_i = r$) annihilate the outer $n-1-i$ rotations of the rightmost yet untreated sequence of descending rotations by a similarity transformation;
- (if $\mathbf{p}_i = \ell$) annihilate the outer $n-1-i$ rotations of the leftmost yet untreated sequence of ascending rotations by a similarity transformation.

The outcome of these $n-2$ steps results in a similar matrix $\hat{A} = \hat{Q}\hat{R}$, where the factorization of $\hat{Q}$ in rotations obeys the pattern identified by $\mathbf{p}$.

There are two building blocks in the design of this algorithm: annihilation of rotations on the left or either on the right. First, in sections 3.2.1 and 3.2.2 a single left and right annihilation step is considered on a pyramid pattern coming from the

[2]Another interpretation could lead to a factorization $P = G_1G_3G_5G_4G_2$, which is, however, owing to commutativity equivalent to $P = G_5G_3G_1G_2G_4$.

[3]Of course, in case a matrix is already structured, some transformations need not be performed. Assume therefore a generic full pyramid shaped factorization of the matrix Q.

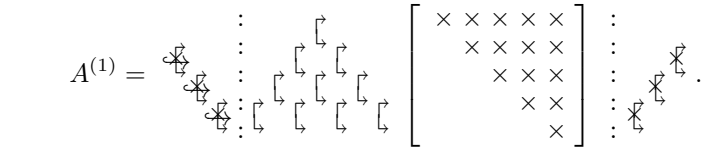
QR-factorization of a 5×5 matrix. The combination of several steps is presented in section 3.2.3.

3.2.1. Left annihilation of a sequence. Consider $A = A^{(0)} = Q^{(0)}R^{(0)}$, factored as in (2.2). Assume $\mathbf{p}_1 = \ell$, requiring the annihilation of the three leftmost iterations. The product of these three rotations $G_1^{(0)}G_2^{(0)}G_3^{(0)}$ in the pyramid shaped factorization of $Q^{(0)}$ defines the similarity transformation

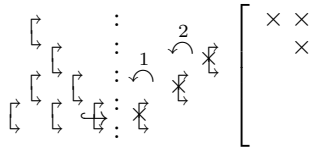
$$A^{(1)} = {G_3^{(0)}}^H {G_2^{(0)}}^H {G_1^{(0)}}^H \left(Q^{(0)}R^{(0)}\right) G_1^{(0)}G_2^{(0)}G_3^{(0)} = Q^{(1)}R^{(1)}, \tag{3.2}$$

where $Q^{(1)}R^{(1)}$ is the QR-factorization of $A^{(1)}$. Note that $Q^{(1)}$ must admit a rotational factorization without the three leftmost rotations and with no extra rotations on the right remaining.

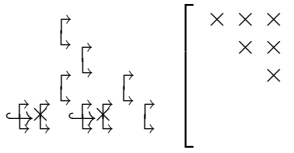
The new QR-factorization of $A^{(1)} = Q^{(1)}R^{(1)}$ is computed as follows. The rotations determining the similarity transformation are marked by $\times$ and separated from the QR-factorization by a vertical dashed line. In the graphical representation (3.3) formula (3.2) is depicted.

$A^{(1)} =$  (3.3)

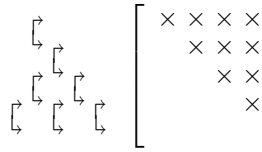
Since the marked rotations are constructed to annihilate three rotations in the left of the pyramid shape, six rotations in total will vanish. The rotations on the right are transferred through the upper triangular matrix $R^{(0)}$ by Lemma 2.6:

$A^{(1)} =$  (3.4)

In scheme (3.4) three more undesired rotations (marked by $\times$) are present. The bottom rotation can be incorporated in the rotation preceding it by a fusion. The remaining two rotations are moved downwards by, respectively, a single and a double shift-through operation. The result is a QR-factorization of the form

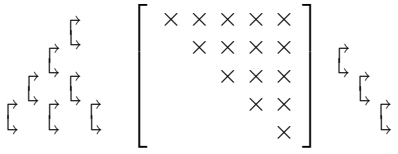
$A^{(1)} =$  (3.5)

Two more fusions in scheme (3.5) are required to obtain the desired rotational factorization of the matrix $Q^{(1)}$, which has now three rotations less than $Q^{(0)}$. The QR-factorization of $A^{(1)} = Q^{(1)}R^{(1)}$ equals

$A^{(1)} =$ 

Important to observe is that the top rotation of the pyramid remains unaltered (it does not even change) during the whole process. We will see that also in the right annihilation process the top rotation remains untouched.

3.2.2. Right annihilation of a sequence. Take $A^{(0)} = Q^{(0)}R^{(0)}$ again as in (2.2). Now $\mathbf{p}_1 = r$, implying the removal of three rotations on the right. Whereas in the left annihilation step the rotations determining the similarity transformation are readily available, this is not the case here. A preliminary step needs to be performed, transferring the rotations designated for removal to the right of the upper triangular matrix. By Lemma 2.6, scheme (2.2) is refactored and thereby transformed in $A^{(0)} = Q^{(0)}R^{(0)} = \tilde{Q}^{(0)}\tilde{R}^{(0)}{G_3^{(0)}}^H{G_2^{(0)}}^H{G_1^{(0)}}^H$.

(3.6) $A^{(0)} =$ 

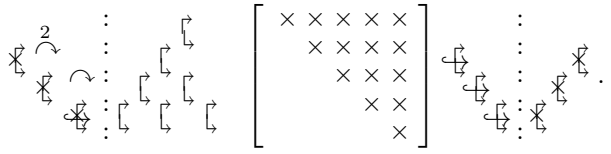
The three rotations on the right ${G_3^{(0)}}^H{G_2^{(0)}}^H{G_1^{(0)}}^H$ determine the similarity transformation

$$A^{(1)} = {G_3^{(0)}}^H{G_2^{(0)}}^H{G_1^{(0)}}^H \left(Q^{(0)}R^{(0)}\right) G_1^{(0)}G_2^{(0)}G_3^{(0)} \tag{3.7}$$

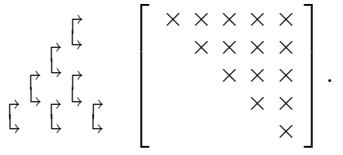
$$= {G_3^{(0)}}^H{G_2^{(0)}}^H{G_1^{(0)}}^H \left(\tilde{Q}^{(0)}\tilde{R}^{(0)}{G_3^{(0)}}^H{G_2^{(0)}}^H{G_1^{(0)}}^H\right) G_1^{(0)}G_2^{(0)}G_3^{(0)} \tag{3.8}$$

$$= {G_3^{(0)}}^H{G_2^{(0)}}^H{G_1^{(0)}}^H \left(\tilde{Q}^{(0)}\tilde{R}^{(0)}\right) = Q^{(1)}R^{(1)},$$

where $Q^{(1)}$ must admit now a rotational factorization without the three rightmost rotations. The operations are quite similar to the ones for a left annihilation step and provide us the new rotational QR-factorization of $A^{(1)} = Q^{(1)}R^{(1)}$. Schematically the following equation is obtained corresponding to factorization (3.8):

$A^{(1)} =$ 

The six rightmost rotations vanish after the fusions. The undesired rotations on the left are dealt with similarly as in section 3.2.1: after a fusion, some shift-through operations, and two more fusions, they disappear and as a result the new factorization of $A^{(1)} = Q^{(1)}R^{(1)}$ is retrieved, with $Q^{(1)}$ satisfying the constraints

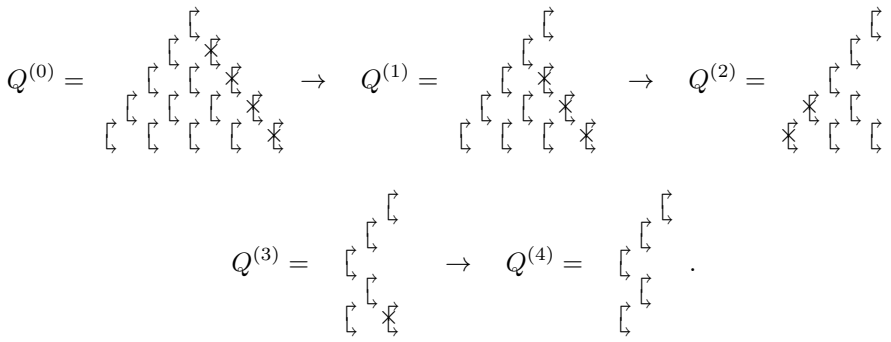
$A^{(1)} =$ 

Again we note that the rotation on top of the pyramid is not altered by any of the performed operations.

3.2.3. Generic algorithm. The generic algorithm is illustrated by a simple example. The clue is that after each annihilation step the top rotation stays in place. Not considering this top rotation leaves us with a new pyramid of height[4] diminished by one, on which we can again apply one of the two annihilation steps described before. Once more the top of the smaller pyramid remains and the procedure continues.

A formal proof of the algorithm producing the exact desired output is done by induction. For a left annihilation in step i, the smaller pyramid is located on the right of the remaining top rotation. Hence, the rotation G_i is positioned on the left of the top rotation G_{i+1} of the smaller pyramid. Note that the position is fixed now, since G_{i+1} will stay in place. For a right annihilation obviously G_i will be positioned on the right of the next remaining rotation G_{i+1}. Hence, after $n-2$ steps, the order of the rotations is fixed and the requested outcome is obtained.

Let us consider, for example, a 6×6 matrix and a reduction process specified by the vector $\mathbf{p} = [r, r, l, r]$. For simplicity, only the rotational factorizations of the unitary matrices $Q^{(i)}$ are depicted. In each step i the rotations to be removed (specified by $\mathbf{p}_i$) are marked by an $\times$.



As a result we obtain a QR-factorization $A^{(4)} = Q^{(4)}R^{(4)}$, where $Q^{(4)}$ is factored by $n-1=5$ rotations satisfying the desired pattern.

4. A double Hessenberg factorization of a compressed unitary matrix. Section 6 presents an alternative type of QR-algorithm suitable for any type of matrix $A = QR$ with Q in compressed format. To make the description of the algorithm comprehensible and compact, two particular factorizations of a zigzag pattern in two Hessenberg matrices will be introduced.

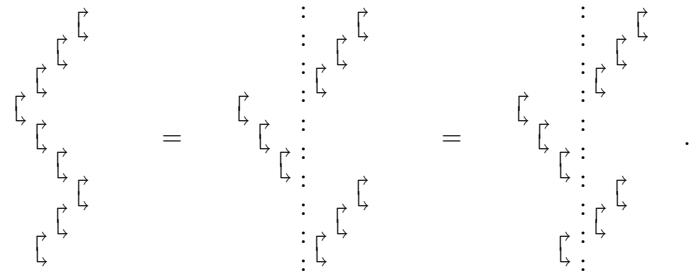
A compressed unitary matrix Q admits a rotational factorization decomposed in two factors Q_d and Q_a: $Q = Q_d Q_a$, where Q_a consists of a single sequence of ascending rotations and Q_d consists of a single sequence of descending rotations.

The mutual position between two successive rotations G_i and G_{i+1} regulates the factorization. The unitary matrix Q_d contains the rotation G_i if rotation G_i precedes G_{i+1} (G_i is located on the left of G_{i+1}; hence $\mathbf{p}_i = \ell$), otherwise G_i is stored in Q_a. Since the trailing rotation G_{n-1} has no successor, it can be put in either unitary matrix.

Example 4.1. Consider the unitary matrix Q in (4.1) with the given zigzag pattern of rotations. The two possible factorizations, depending on the position of the trailing rotation, of Q are shown. The rotations belonging to Q_d are on the left of the dashed line; the rotations of Q_a are on the right. Based on commutation properties

[4]The height of pyramid agrees with the dimension of the associated matrix.

the rotations can be slightly reordered such that visually a single descending and a single ascending sequence of rotations is obtained:

(4.1) $Q =$  .

Another example of a *descending-ascending (DA-)factorization* is the CMV-decomposition (see (3.1)).

In matrix terms this factorization states that every compressed unitary matrix can be factored in a product of a unitary upper Hessenberg and a unitary lower Hessenberg matrix, or briefly a double Hessenberg factorization. More precisely, the upper and lower Hessenberg matrices consist of nonoverlapping diagonal blocks of irreducible Hessenberg matrices. Comparing, however, the blocks of the lower and the upper Hessenberg matrices, interaction between the top and bottom rows of these blocks takes place. This factorization can be seen as a generalization of the CMV-decomposition towards diagonal blocks of sizes exceeding 2×2.

Remark 4.2. Any compressed unitary matrix admits also an ascending-descending (AD-)factorization $Q = Q_a Q_d$. The unitary matrix Q_a contains the rotation G_i if rotation G_i precedes G_{i-1} (G_i is positioned on the left of G_{i-1}). Otherwise G_i is stored in Q_d. Since the first rotation G_1 has no predecessor, it can be put in either unitary matrix. This is a factorization of a compressed unitary matrix in a product of a unitary lower Hessenberg and a unitary upper Hessenberg matrix.

Remark 4.3. The QR-type algorithm presented in section 6 relies on the DA-factorization, but also the AD-factorization can be used without any essential changes taking place. For example, the chasing technique (see section 7) based on a DA-factorization goes from top to bottom and based on a DA-factorization it runs from bottom to top. Up to a certain sense the links are similar to the ones between GR-type and RG-type iterations.

There is another intriguing link with eigenvalue computations. Consider unitary matrices. The algorithm proposed in [2] transforms a regular or generalized eigenvalue problem involving unitary matrices to a generalized eigenvalue problem, where both matrices are block diagonal matrices having 2×2 unitary blocks on the diagonal. The structure of these matrices coincides with the one from the matrices U and V in the CMV-decomposition of a pentadiagonal unitary matrix $P = UV$. Taking now advantage of the DA-factorization, we can generalize this and get a generalized eigenvalue problem where both matrices are unitary upper Hessenberg matrices. These unitary upper Hessenberg matrices inherit the structure from the factors in the DA-factorization, and consist therefore of overlapping blocks of irreducible Hessenberg matrices. Both the pencil in [2] and the pencil of irreducible Hessenberg matrices use the same number of parameters, namely $n - 1$ rotations.

To avoid the cumbersome and even misleading wording "QR-type method," we refer to the next iterative algorithm for computing the eigenvalues as the *DA-algorithm.*

5. Uniqueness of the similarity transformation. An important, yet unanswered question is the one of uniqueness. The theorem formulated in this section is sort of an extension of the implicit Q-theorem, since also the Hessenberg case is covered. The proof provided here is based on Krylov matrices as in [36], which is more appealing than direct calculations as in [17, 27] for Hessenberg(-like) matrices. As mentioned before, for simplicity, we restrict ourselves in this article to nonsingular matrices.

THEOREM 5.1 (implicit Q-theorem). *Consider a nonsingular matrix A and a position vector $\mathbf{p}$. Let V_1 and V_2 be two unitary matrices sharing the same first column (up to a unimodular factor) such that*

$$Q_1 R_1 = A_1 = V_1^H A V_1 \quad \text{and} \quad Q_2 R_2 = A_2 = V_2^H A V_2,$$

where the unitary factors Q_1 and Q_2 in the QR-decompositions of A_1 and A_2 obey the pattern specified by $\mathbf{p}$ and, in addition, all rotations appearing in these factorizations of Q_1 and Q_2 differ from the identity.[5] *Hence the matrices A_1 and A_2 are essentially identical.*

The zigzag patterns arising in the various factorizations of compressed unitary matrices have a close relation to the ordering of monomials when considering the recurrence relations for orthogonal Laurent polynomials [4]. Here the ordering of multiplication with A or A^{-1} will play a key role in the construction of the involved rational Krylov matrices.

We consider rational Krylov spaces $\mathcal{K}_{\mathbf{p},k}(A, \mathbf{v})$ of length k with starting vector $\mathbf{v}$ spanned by k vectors out of the sequence

$$\ldots,\ A^3\mathbf{v},\ A^2\mathbf{v},\ A\mathbf{v},\ \mathbf{v},\ A^{-1}\mathbf{v},\ A^{-2}\mathbf{v}, A^{-3}\mathbf{v},\ \ldots. \tag{5.1}$$

The order in which the vectors appear in the sequence is determined by the position vector $\mathbf{p}$. The first vector in the Krylov space is always the middle vector $\mathbf{v}$. If $\mathbf{p}_i = \ell$, the next vector on the left is taken to fill up position $i+1$; if $\mathbf{p}_i = r$, one takes the next one on the right. Since $\mathbf{p}_{n-1}$ is not specified, the last vector—the one in the nth position—can either be taken from the left or from the right; this is an optional choice.[6] The associated Krylov matrix $K_{\mathbf{p},k}(A, \mathbf{v})$ has these just determined vectors as columns of an $n \times k$ matrix. When $k = n$, k is omitted as subscript in $\mathcal{K}$ and K.

For example, the CMV-pattern in scheme (3.1) is determined by $\mathbf{p} = [\ell, r, \ell, r]$. The corresponding rational Krylov space is $\mathcal{K}_{\mathbf{p},5}(A, \mathbf{v}) = \text{span}\{\mathbf{v},\ A\mathbf{v},\ A^{-1}\mathbf{v},\ A^2\mathbf{v},\ A^{-2}\mathbf{v}\}$. Since $n = 6$, the space $\mathcal{K}_{\mathbf{p},6}(A, \mathbf{v})$ equals $\text{span}\{\mathbf{v},\ A\mathbf{v},\ A^{-1}\mathbf{v},\ A^2\mathbf{v},\ A^{-2}\mathbf{v},\ A^3\mathbf{v}\}$ or $\text{span}\{\mathbf{v},\ A\mathbf{v},\ A^{-1}\mathbf{v},\ A^2\mathbf{v},\ A^{-2}\mathbf{v},\ A^{-3}\mathbf{v}\}$.

The proof of Theorem 5.1 is decomposed in two parts. Consider a matrix A reduced via the similarity transformation from section 3 to a matrix factorization $V^H A V = \hat{A} = \hat{Q}\hat{R}$ according to a position vector $\mathbf{p}$. Crucial in the proof of Theorem 5.1 is the observation that $K_{\mathbf{p}}(\hat{A}, \mathbf{e}_1)$ is upper triangular, which forms the first and most technical part of the proof (see subsection 5.1). Once this is known, the second part proceeds identical as in [36] (see subsection 5.2).

5.1. Upper triangular Krylov matrix. All rotations in the rotational factorization of the compressed unitary matrix $\hat{Q}$ are required to be different from the

[5] This is related to irreducibility and we will come back to this in section 6.1. When an identity rotation is encountered, uniqueness is only guaranteed up to this position, just like the standard implicit Q-theorem.

[6] We point out that freedom of the trailing items is recurring throughout the article.

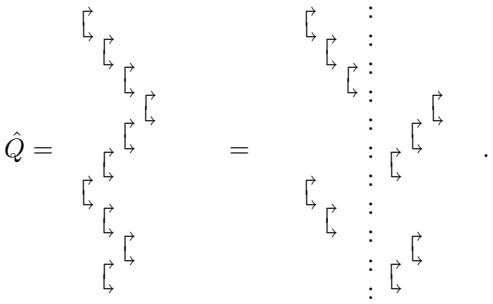
identity. Otherwise the proof will break down and essential uniqueness can only be guaranteed up to a certain point. To prove that $K_{\mathbf{p}}(\hat{A}, \mathbf{e}_1)$ is upper triangular the DA-factorization of $\hat{Q} = \hat{Q}_d \hat{Q}_a$ is utilized. Both $\hat{Q}_d$ and $\hat{Q}_a$ are built up by nonoverlapping diagonal blocks, in which each block contains a sequence of rotations (see (4.1)); the location of these blocks is vital for the proof.

We will investigate the zero-structure of the vectors $\hat{A}^k \mathbf{e}_1$, using the DA-decomposition of $\hat{Q}$. The analysis of $\hat{A}^{-k}\mathbf{e}_1$ proceeds similarly, since $\hat{Q}^{-1} = \hat{Q}_a^{-1}\hat{Q}_d^{-1}$ is a DA-decomposition of $\hat{Q}^{-1}$, and by Lemma 2.6 it is known that the DA-decomposition of the Q-factor of the QR-factorization of $\hat{A}^{-1}$ has the same pattern as the DA-decomposition $\hat{Q}_a^{-1}\hat{Q}_d^{-1}$.

As before, the vectors $\mathbf{e}_i$ denote the standard basis vectors; the vectors $\tilde{\mathbf{e}}_i$ stand for vectors having the element in position i different from zero, the elements below i are zero, and the elements above might be nonzero. In fact $\tilde{\mathbf{e}}_i$ equals a linear combination of the $\mathbf{e}_j$, $1 \leq j \leq i$, where the coefficient of $\mathbf{e}_i$ is nonzero; the other coefficients are unspecified.

The study of the structure of $K_{\mathbf{p}}(\hat{A}, \mathbf{e}_1)$ is quite technical; therefore the flow of the proof will be accompanied by an example in which $\mathbf{p} = [\ell, \ell, \ell, r, r, r, \ell, \ell, r]$. A DA-decomposition of $\hat{Q}$ corresponding to the example is of the form

(5.2)



The indices i_j are defined as follows: $i_1 = 1$ and i_j is the position k such that the $(j-1)$th transition from ℓ to r or r to ℓ takes place between $\mathbf{p}_{k-1}$ and $\mathbf{p}_k$. The list is closed by a trailing $i_j = n$, where n is the dimension of $\hat{Q}$. In the example $i_1 = 1, i_2 = 4, i_3 = 7, i_4 = 9$, and $i_5 = 11$. For simplicity we assume that $\mathbf{p}_1 = \ell$; if it equals r, only a change between even and odd in the upcoming part is required. Based on this indexing, we can identify diagonal blocks in the matrices $\hat{Q}_a$ and $\hat{Q}_d$: the submatrices[7] $\hat{Q}_d(i_j : i_{j+1})$ (for odd j) are of Hessenberg form; the submatrices $\hat{Q}_a(i_j : i_{j+1})$ (for even j) are of lower Hessenberg form; the remaining diagonal blocks in $\hat{Q}_a$ and $\hat{Q}_d$ equal the identity. Please note that for both matrices $\hat{Q}_d$ and $\hat{Q}_a$ the elements in the positions (i_j, i_j) are included in a Hessenberg (for $\hat{Q}_d$) and a lower Hessenberg structure (for $\hat{Q}_a$).

The derivations in the next part are only valid when all rotations are nonidentical. Based on the DA-factorization and the index set, one can verify that (thereby using several times Lemma 2.6)

$$\hat{A}^k \mathbf{e}_1 = \tilde{\mathbf{e}}_{k+1} \quad \text{for } i_1 \leq k \leq i_2 - 1,$$

since the entire block $\hat{Q}_a(i_1 : i_2 - 1)$ equals the identity.

[7]Since only square subblocks are considered, the notation $Q(i : j)$ is a shorthand for MATLAB style notation $Q(i : j, i : j)$.

Considering, however, $\hat{A}^{i_2}$, the matrix $\hat{Q}_a$ will come into play since it has a lower Hessenberg block on the diagonal, ranging from position i_2 to i_3. This interaction with $\hat{Q}_a$ will smear out nonzero elements promptly until position i_3. This means that we get (there is a little abuse in notation by considering $\hat{R}\tilde{\mathbf{e}}_{i_2} = \tilde{\mathbf{e}}_{i_2}$, but in the spirit of the undefinedness of some elements in the vector $\tilde{\mathbf{e}}_{i_2}$ this is appropriate)

$$\hat{A}^{i_2}\mathbf{e}_1 = \hat{A}\ \tilde{\mathbf{e}}_{i_2} = \hat{Q}_d\ \hat{Q}_a\ \hat{R}\ \tilde{\mathbf{e}}_{i_2} = \hat{Q}_d\ \hat{Q}_a\ \tilde{\mathbf{e}}_{i_2} = \hat{Q}_d\tilde{\mathbf{e}}_{i_3} = \tilde{\mathbf{e}}_{i_3+1}.$$

The action of $\hat{Q}_d$ on $\tilde{\mathbf{e}}_{i_3}$ is properly defined because of the Hessenberg block $\hat{Q}_d(i_3 : i_4)$. Roughly speaking one can say that the diagonal block in $\hat{Q}_a$ shifts down the nonzero elements quite dramatically until position i_3. From this position there is temporarily again no impact of $\hat{Q}_a$, signifying that the Hessenberg structure of $\hat{Q}_d$ can again shift down the nonzero elements in the vector one by one. The Hessenberg part in action is now $\hat{Q}_d(i_3 : i_4)$. Yet, once position i_4 is reached, $\hat{Q}_a$ comes in action again. Combining all this information the following formulas are retrieved:

$$\hat{A}^{k-i_3+i_2}\mathbf{e}_1 = \tilde{\mathbf{e}}_{k+1} \quad \text{for } i_3 \leq k \leq i_4 - 1,$$

indicating that the powers from $\hat{A}$ continue to grow from i_2, but the nonzero elements have shifted down extra positions until the next i_j for j odd.

Putting all things together the following formula predicts the nonzero-structure of the powers of $\hat{A}$ applied on $\mathbf{e}_1$ (take $i_0 = 1$):

$$\hat{A}^{k+\sum_{\hat{\jmath}\ odd\ \&\ \hat{\jmath}\leq j}(i_{\hat{\jmath}-1}-i_{\hat{\jmath}})}\ \mathbf{e}_1 = \tilde{\mathbf{e}}_{k+1} \quad \text{for } i_j \leq k \leq i_{j+1} - 1, \text{ where } j \text{ is odd.}$$

In words this simply states the following: when an i_j is reached for even j, there is a downward shift of the nonzeros up to position i_{j+1}. The somehow complicated notation of the power just expresses that the power keeps increasing by one each step. For our particular example the following relations are obtained:

$$\hat{A}\mathbf{e}_1 = \tilde{\mathbf{e}}_2, \quad \hat{A}^2\mathbf{e}_1 = \tilde{\mathbf{e}}_3, \quad \hat{A}^3\mathbf{e}_1 = \tilde{\mathbf{e}}_4, \quad \hat{A}^4\mathbf{e}_1 = \tilde{\mathbf{e}}_8, \quad \hat{A}^5\mathbf{e}_1 = \tilde{\mathbf{e}}_9, \quad \text{and} \quad \hat{A}^6\mathbf{e}_1 = \tilde{\mathbf{e}}_{11}.$$

For the inverse powers $\hat{A}^{-k}$ similar formulas are acquired:

$$\hat{A}^{-(k+\sum_{\hat{\jmath}\ even\ \&\ \hat{\jmath}\leq j}(i_{\hat{\jmath}-1}-i_{\hat{\jmath}}))}\ \mathbf{e}_1 = \tilde{\mathbf{e}}_{k+1} \quad \text{for } i_j \leq k \leq i_{j+1} - 1, \text{ where } j \text{ is even.}$$

The example leads to the following equalities:

$$\hat{A}^{-1}\ \mathbf{e}_1 = \tilde{\mathbf{e}}_5, \quad \hat{A}^{-2}\ \mathbf{e}_1 = \tilde{\mathbf{e}}_6, \quad \hat{A}^{-3}\ \mathbf{e}_1 = \tilde{\mathbf{e}}_7, \quad \hat{A}^{-4}\ \mathbf{e}_1 = \tilde{\mathbf{e}}_{10}, \quad \text{and} \quad \hat{A}^{-5}\ \mathbf{e}_1 = \tilde{\mathbf{e}}_{11}.$$

Gluing all the results for the powers and inverse powers together, taking thereby the ordering of the position vector $\mathbf{p}$ into account, one gets $K_{\mathbf{p},k}(\hat{A}) = [\tilde{\mathbf{e}}_1, \ldots, \tilde{\mathbf{e}}_k]$, which is obviously upper triangular.

Indeed testing it on our example we retrieve

$$\begin{aligned} K_{\mathbf{p}}(\hat{A}, \mathbf{e}_1) &= [\mathbf{e}_1,\ A\mathbf{e}_1,\ A^2\mathbf{e}_1,\ A^3\mathbf{e}_1,\ A^{-1}\mathbf{e}_1,\ A^{-2}\mathbf{e}_1,\ A^{-3}\mathbf{e}_1,\ A^4\mathbf{e}_1,\ A^5\mathbf{e}_1,\ A^{-4}\mathbf{e}_1,\ A^{-5}\mathbf{e}_1] \\ &= [\mathbf{e}_1,\ A\mathbf{e}_1,\ A^2\mathbf{e}_1,\ A^3\mathbf{e}_1,\ A^{-1}\mathbf{e}_1,\ A^{-2}\mathbf{e}_1,\ A^{-3}\mathbf{e}_1,\ A^4\mathbf{e}_1,\ A^5\mathbf{e}_1,\ A^{-4}\mathbf{e}_1,\ A^6\mathbf{e}_1] \\ &= [\tilde{\mathbf{e}}_1,\ \ldots,\ \tilde{\mathbf{e}}_{11}]. \end{aligned}$$

This concludes the technical part for proving that $K_{\mathbf{p}}(\hat{A}, \mathbf{e}_1)$ is upper triangular.

5.2. Uniqueness based on the *QR*-factorization. By the definition of the rational Krylov matrices and using $\hat{A} = V^H AV$ we have

$$VK_{\mathbf{p}}(\hat{A}, \mathbf{e}_1) = K_{\mathbf{p}}(V\hat{A}V^H, V\mathbf{e}_1) = K_{\mathbf{p}}(A, V\mathbf{e}_1). \tag{5.3}$$

Using the notation of Theorem 5.1 the following equalities are obtained (σ is a unimodular factor):

$$\begin{aligned} V_1 K_{\mathbf{p}}(A_1, \mathbf{e}_1) = K_{\mathbf{p}}(V_1 A_1 V_1^H, V_1\mathbf{e}_1) &= K_{\mathbf{p}}(A, V_1\mathbf{e}_1) \\ &= \sigma K_{\mathbf{p}}(A, V_2\mathbf{e}_1) \\ &= \sigma K_{\mathbf{p}}(V_2 A_2 V_2^H, V_2\mathbf{e}_1) = \sigma V_2 K_{\mathbf{p}}(A_2, \mathbf{e}_1). \end{aligned}$$

The left and the right terms are QR-factorizations. The essential uniqueness of the QR-factorization implies that V_1 and V_2 are essentially identical proving thereby Theorem 5.1.

We will not elaborate on this, but the above relations show that the corresponding Krylov spaces uniquely determine the similarity transformation to bring A to $\hat{A}$ format. This indicates that the Q-factor from the QR-factorization of the rational Krylov matrix is also suited to unitarily transform the matrix to the desired format.

6. Eigenvalue computations. Given a matrix A and a suitable shift μ chosen to enhance the convergence, a single shifted QR-step is determined by two equations:

$$A - \mu I = QR, \tag{6.1}$$

$$\hat{A} = RQ + \mu I, \tag{6.2}$$

where I stands for the identity matrix and $\hat{A}$ is the outcome and new iterate. Executing (6.1) and (6.2) corresponds to the explicit version of the QR-algorithm, where the matrices Q and R are really computed explicitly before the reverse product is calculated. The implicit version determines the unitary matrix Q on the fly and performs the following transition, without ever explicitly forming the unitary matrix:

$$\hat{A} = Q^H AQ. \tag{6.3}$$

One can also define the matrix $\hat{A}$ as

$$\hat{A} = RAR^{-1}; \tag{6.4}$$

this formula is, however, merely of theoretical interest for proving, for instance, the preservation of the structure of the matrix A.

6.1. Irreducibility. A matrix A with QR-factorization $A = QR$, where the unitary matrix Q is in compressed format, is said to be irreducible if and only if the upper triangular matrix has all diagonal elements, except the trailing one, different from zero and if the rotations involved in the factorization of the unitary matrix Q differ from the identity. This definition is equivalent with the standard definitions of irreducibility of Hessenberg and Hessenberg-like matrices. A rotation equal to the identity implies that the matrix is block upper triangular and as such can be split in submatrices for computing the eigenvalues.

6.2. Explicit version of the new iteration. Consider a matrix A with a QR-factorization having the unitary matrix in compressed form and DA-factored: $A = Q_d\tilde{Q}_a\tilde{R}_\ell$. The shifted DA-iteration proposed here consists of the following steps. First, the ascending sequence of rotations $\tilde{Q}_a$ is brought to the right of the upper triangular matrix $\tilde{R}_\ell$ by Lemma 2.6: $\tilde{Q}_a\tilde{R}_\ell = R_rQ_a$. For a shifted matrix A we have

$$A - \mu I = Q_d\tilde{Q}_a\tilde{R}_\ell - \mu I = \left(Q_dR_r - \mu Q_a^H\right) Q_a = QRQ_a, \tag{6.5}$$

where the product QR represents the QR-factorization of the term between the brackets. This unitary matrix Q determines the similarity transformation of the DA-step. The new iterate is of the form $\hat{A} = Q^HAQ$. In a certain sense this is sort of a URV-factorization of the matrix $A - \mu I$, where U determines the next similarity transformation. We remark that the DA-factorization admits two variants, depending on the position of the final rotation. As a result there are also two variants for executing a DA-step.

Let us elaborate a little on the structure of Q. The number of rotations appearing in the rotational factorization of the unitary matrix Q is essential for keeping the computational complexity under control; here Q admits a factorization in a single descending sequence of rotations, since $Q_dR_r - \mu Q_a^H$ is an irreducible Hessenberg matrix. Descending sequences of rotations applied on an upper triangular matrix result in a Hessenberg matrix. Hence, two Hessenberg matrices are obtained: $H_\ell = Q_dR_r$ and $H_r = \mu Q_a^H$. Based on the diagonal blocks in the matrices Q_a and Q_d, one can partition the matrices H_ℓ and H_r. It is easily verified that the diagonal blocks of upper triangular form in H_ℓ correspond with diagonal blocks of Hessenberg form in H_r and vice versa. Therefore, the summation of both Hessenberg matrices leads to a Hessenberg matrix $H = H_\ell + H_r$. In case A is irreducible and μ is no eigenvalue of A, the Hessenberg matrix is irreducible as well, admitting thus an essentially unique QR-factorization composed of $n-1$ rotations.

Before proceeding with the analysis of this iteration, the formulas equivalent to (6.1), (6.3), and (6.4) are presented. Straightforward calculations imply the following relations, using thereby (6.5):

$$\hat{A} = (RQ_a)Q + \mu I, \quad \hat{A} = Q^HAQ, \quad \text{and} \quad \hat{A} = RQ_aAQ_a^HR^{-1}. \tag{6.6}$$

Both the QR-algorithm for Hessenberg matrices and the rational driven QR-iteration for Hessenberg-like matrices fit nicely in this framework. Consider for example a Hessenberg matrix $H = \tilde{Q}\tilde{R}$. Take the DA-decomposition $Q_d = \tilde{Q}$ and $\tilde{Q}_a = I$, implying that the final rotation is incorporated in the descending sequence. Reconsidering (6.5) for the Hessenberg matrix H gives us $\tilde{R}_\ell = \tilde{R}$ leading to $A - \mu I = \tilde{Q}\tilde{R} - \mu I = QR$. As a result $\hat{A} = Q^HAQ$, which is a step of the standard QR-algorithm. For a Hessenberg-like matrix $A = \tilde{Q}\tilde{R}$, the matrix $\tilde{Q}$ is an ascending sequence of rotations. The DA-decomposition considered is $\tilde{Q} = I\tilde{Q}_a$, with the final rotation put in the ascending part of the factorization. Rewriting the formulas from (6.5) gives us $\tilde{R}_\ell = \tilde{R}$ and thus $A - \mu I = I\tilde{Q}_a\tilde{R}_\ell - \mu I = \left(R_r - \mu Q_a^H\right) Q_a = QRQ_a$. The new iterate $\hat{A}$ corresponds to a step of the QH-algorithm [30, 31] or the rational driven QR-iteration on the matrix A.

6.3. Structure of the rotational factorization pattern after an iteration. One of the crucial features of the QR-iteration applied on Hessenberg and Hessenberg-like matrices is the preservation of the structure. Consider, however, a nonsingular matrix $A = QR$ with Q in compressed format. Based on (6.4) and Lemma 2.6 it is

trivial to prove that the patterns in the factorization of the matrix Q are preserved under QR-steps. For a general treatment of structures preserved under QR-iterates we refer to [12]; the singular case is treated thoroughly in [11].

We did not focus on the development of QR-algorithms for these various types of patterns because the number of rotations needed for determining the similarity transformation is too high. Executing, for example, a step of the traditional QR-method on a Hessenberg-like matrix involves the computation of $2n-3$ rotations. A single shifted QR-step for a Hessenberg matrix only needs $n-1$ rotations. One can roughly claim that for dealing with an ascending sequence of rotations the double amount of rotations in a QR-step is entailed. Another reason is that these matrices belong to the more general class of quasi-separable matrices (both Hessenberg and Hessenberg-like matrices can be considered as quasi-separable matrices). For quasi-separable matrices explicit QR-algorithms, determined by much more than $n-1$ rotations each step, are readily available [10, 15, 28].

The DA-algorithm proposed here is not precisely the same as the QR-method; hence we cannot rely on the structure preserving theorems from [12] and [11]. It will be shown that generically the matrix structure and the shape of the rotations are not preserved under an iteration, but alter after each step. We are particularly interested in the shape of the rotational factorization of the matrix $\hat{Q}$ in the QR-factorization of the matrix $\hat{A} = \hat{Q}\hat{R}$ as the result of a DA-step, since this fully determines the matrix structure. Even though the patterns might vary, we will see that the Q-factors remain in compressed format.

Given a matrix A, the result $\hat{A}$ after a step of the DA-iteration is of the form $\hat{A} = RQ_aAQ_a^HR^{-1}$. Using Lemma 2.6, we know that the upper triangular matrices R and R^{-1} have no impact on the pattern of the Q-factor in the QR-factorization of the matrix $\hat{A}$. It remains to investigate the structure of $Q_aAQ_a^H$. Using $A = Q_dR_rQ_a$, we get that the Q-factor in the factorization of $\hat{A}$ shares the pattern of Q_aQ_d.

So, the original pattern in the rotational QR-factorization of A equals the pattern of Q_dQ_a, whereas the new pattern coming from $\hat{A}$ is equivalent to the pattern of Q_aQ_d. A close look at the reverting of the order reveals that the pattern moves up one position (consider, for example, (4.1) and an example is given in (7.5)), with the position of the last rotation determined by the variant of the DA-factorization taken to determine the DA-step. The flexibility of the position of the last rotation in the decomposition of Q_dQ_a implies that one can alter the pattern on the fly, since both choices determine also slightly different resulting structures. This flexibility of the final rotation in the DA-factorization appears also in the final column of the Krylov matrices and in the position of the final rotation in the structure preservation. Moreover, it will again pop up when discussing the implicit version of the DA-algorithm.

7. Implicit version of the new iteration. The most elegant form for executing steps of the QR-algorithm on a Hessenberg matrix performs these steps implicitly. Instead of computing the entire sequence of rotations determining the similarity transformation, the first rotation is sufficient (for a single shifted step) and the remaining rotations are determined on the fly. Applying the similarity transformation determined by this rotation on the matrix perturbs its structure in the upper left corner (*initialization step*). Since the final structure after the similarity transformation is known beforehand one can apply a sequence of $n-2$ structure restoring transformations (*chasing steps*) such that the resulting matrix meets the structural constraints. Accomplishing a QR-step in this manner does hence not require the computation of the complete unitary matrix, determining the similarity transform, in advance.

Implicit QR-algorithms for Hessenberg(-like) matrices are common knowledge [31, 36, 38, 40]. The most popular method is the bulge chasing method, where the elements perturbing the Hessenberg structure are chased downwards until they slide off the matrix. When exploiting the rotational QR-factorization for studying the bulge chasing algorithm, a rotation chasing method is obtained. In this case a perturbing rotation is chased downwards until it vanishes.

Performing a DA-iteration on a zigzag pattern exploits chasing techniques from both ascending (lower Hessenberg) and descending (Hessenberg) sequences. A brief recapitulation of these techniques, followed by a strategy to combine both, is considered next and results in an implicit DA-iteration.

7.1. Purely descending and ascending sequences. Only the single shifted case is considered, implying that the initialization step is composed of a single rotation. To pinpoint the difference between descending and ascending sequences both cases are discussed simultaneously. The major difference lies in the determination of the chasing rotations. In the descending case these rotations pop up on the left side, whereas in the ascending case they appear on the right side of the factorization. The initial rotation is determined by the first column of the Hessenberg matrix $Q_d R_r - \mu Q_a^H$ from (6.5).

7.1.1. Initialization. Consider a Hessenberg matrix H and Hessenberg-like matrix Z factored as in (2.3). The next graphical scheme depicts the initialization step applied on H and Z, where G_1 and $\tilde{G}_1$ are suitably constructed: $H^{(1)} = G_1^H H G_1$ and $Z^{(1)} = \tilde{G}_1^H Z \tilde{G}_1$. The rotations involved in the similarity transformation are marked by an $\times$.

$$H^{(1)} = \qquad \text{and} \qquad Z^{(1)} = .$$

Updating the representation starts by transferring the rotations on the right in both cases through the upper triangular matrix (Lemma 2.6):

$$H^{(1)} = \qquad \text{and} \qquad Z^{(1)} = .$$

Removal of one of the two perturbing rotations is done by a fusion on the left in the Hessenberg case and a fusion on the right of the Q-factor in the Hessenberg-like case. After the fusion, a shift-through operation is performed (to the left for the Hessenberg, to the right for the Hessenberg-like); as a result one obtains the following factorizations:

$$H^{(1)} = \qquad \text{and} \qquad Z^{(1)} = .$$

To finalize this initialization step we transfer the marked rotation in the middle of $Z^{(1)}$ back to the right:

$$Z^{(1)} = \quad .$$

The result of the initialization step gives us for $H^{(1)}$ the QR-factorization of a Hessenberg matrix $H^{(1)} = G_2\left(Q^{(1)}R^{(1)}\right)$, perturbed on the left with a rotation G_2 and the QR-factorization of $Z^{(1)} = \left(\tilde{Q}^{(1)}\tilde{R}^{(1)}\right)\tilde{G}_2^H$, which is the one of a Hessenberg-like matrix perturbed on the right by a rotation $\tilde{G}_2^H$. Both disturbing rotations act on rows 2 and 3.

7.1.2. Chasing the rotations. Chasing the rotations in both cases is quite simple. The following two similarity transformations will chase the perturbing rotation down one position:

$$H^{(2)} = G_2^H H^{(1)} G_2 = Q^{(1)}R^{(1)}G_2, \tag{7.1}$$

$$Z^{(2)} = \tilde{G}_2^H Z^{(1)} \tilde{G}_2 = \tilde{G}_2^H \tilde{Q}^{(1)} \tilde{R}^{(1)}. \tag{7.2}$$

To deal with $H^{(2)}$, first G_2 is transferred through the matrix $R^{(1)}$ to the left and then a shift-through operation is executed resulting again in a matrix factorization $H^{(2)} = G_3 Q^{(2)} R^{(2)}$, where the perturbing rotation G_3 acts now on rows 3 and 4. Dealing with $Z^{(2)}$ proceeds sort of in inverse order. First the shift-through operation to the right is performed, followed by transferring the newly obtained rotation through the upper triangular matrix $\tilde{R}^{(1)}$. The result is a factorization $Z^{(2)} = \tilde{Q}^{(2)}\tilde{R}^{(2)}\tilde{G}_3^H$ with $\tilde{G}_3$ acting on rows 3 and 4. Graphically these transitions look as follows, starting with the factorizations (7.1) and (7.2):

$$H^{(2)} = \quad , \quad Z^{(2)} = \quad ,$$

$$H^{(2)} = \quad , \quad Z^{(2)} = \quad , \tag{7.3}$$

$$H^{(2)} = \quad , \quad Z^{(2)} = \quad . \tag{7.4}$$

After $n-2$ chasing steps one is not able to perform a shift-through operation anymore because the end of the sequence is reached. The perturbing rotation can now be fused with the trailing rotation from the sequence and vanishes. As a result one retrieves again the proper rotational factorizations in a descending or an ascending sequence.

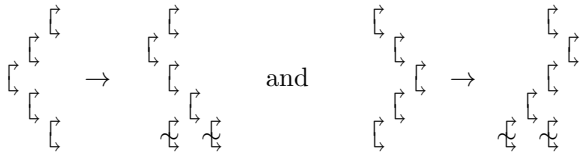
The sequences of rotations considered in the next section are not purely of descending or ascending form. Performing an implicit step on a zigzag pattern will therefore be constituted of both techniques. Executing a similarity transformation to remove an undesired rotation as in the left of (7.4) will be referred to as the *removal of a left rotation*, whereas a similarity transformation and the corresponding steps to get rid of the perturbing rotation in (7.3) will be referred to as the *removal of a right rotation*.

7.2. Zigzag patterns. The result of a DA-step on a QR-factorization of a matrix A, where Q is factored in a zigzag pattern, results in a new upwards shifted pattern (see section 6.3). Based on this knowledge, it is possible to perform a DA-step implicitly. The remaining issue is, however, the transition from descending to ascending sequences and, conversely, from ascending to descending.

A generic zigzag pattern is constituted of parts which are descending, ascending or turns from ascending to descending and vice versa. The previous section explained purely ascending or descending parts. Here focus will be put on the change of direction and on the determination of the final (flexible) rotation. A global implicit version simply combines all the building blocks from the previous subsections and this subsection.

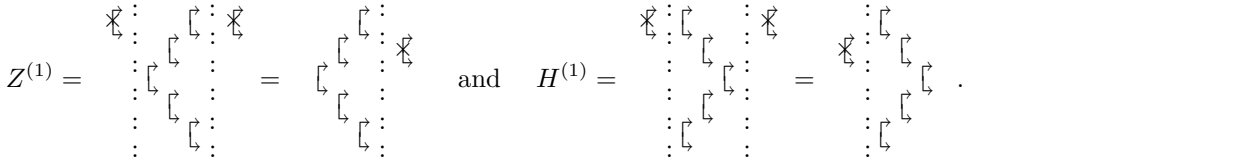
The upward move of the pattern means that for the connections between ascending and descending sequences the patterns on the left of the arrow in (7.5) turn into the patterns on the right of the arrow. The transitions for both <- and >-corners are shown in (7.5). The upward move of the pattern leaves the position of the final rotation undetermined; only one of the rotations marked by the $\sim$ sign (7.5) in a sequence can remain:

(7.5)



From subsection 7.1.2 the removal of either a left or right perturbing rotation is known. In the next graphical flow of the transition from left to right we will omit the upper triangular matrix. This matrix does not complicate matters; only the extra transferring of rotations through the matrix should not be forgotten when implementing the method.

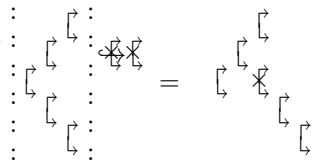
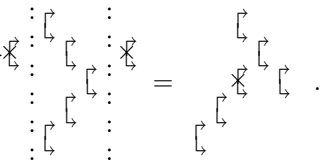
7.2.1. Initialization. The matrix corresponding with a >-pattern is denoted as $H^{(0)}$ and the matrix $Z^{(0)}$ corresponds to the <-pattern. The initial similarity transformations $H^{(1)} = G_1^H H^{(0)} G_1$ and $Z^{(1)} = \tilde{G}_1^H Z^{(0)} \tilde{G}_1$ are carried out just as in subsection 7.1; as result we get



In these figures an extra vertical dashed line is drawn to separate the matrix from its perturbing rotations. For the matrix $Z^{(1)}$ the remaining disturbing rotation appears on the right; for $H^{(1)}$ it is spotted on the left.

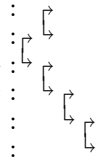
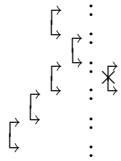
7.2.2. Chasing the rotations. The chasing starts, another time, just as for the purely descending or ascending case. The similarity transformations to obtain $H^{(2)}$ and $Z^{(2)}$ are determined analogously as in subsection 7.1.2. In scheme (7.6) the full similarity transformation is depicted; for $Z^{(2)}$ this is based on the removal of a right rotation, and for $H^{(2)}$ this is based on the removal of a left rotation.

(7.6)

$Z^{(2)} =$  and $H^{(2)} =$  .

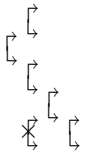
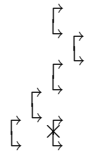
Looking again at the desired result in (7.5), we see that after this step the resulting matrix on the right has a change of direction in both patterns. Indeed, after this similarity transformation (7.6), we notice that the rotations conventionally determining the upcoming chasing step (marked by $\times$) are not *free* anymore, which means that they are blocked by other rotations. This is also reflected in the schemes, as it is impossible to separate the perturbing rotation from the remaining pattern by a single vertical dashed line. The change of direction of the corresponding patterns created a sort of barrier blocking these rotations.

The problem is easily solved by changing the undesired rotation. Since the blocked rotation satisfies the final pattern, this rotation is kept and the other outer rotations are now marked

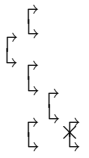
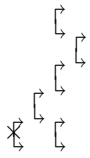
$Z^{(2)} =$  and $H^{(2)} =$  .

In the adapted patterns one can again draw the vertical dashed line disconnecting the unwanted rotation from the others. At this point the order of removal is changed. When the first perturbing rotation was on the right (resp., left) it is now on the left (resp., right). These new undesired rotations can be removed once more by the standard techniques for removing a left or a right perturbing rotation. The process continues by the standard routines for either descending or ascending sequences as put forward in section 7.1.

7.2.3. Options for the final rotation. After the initial step and $n-3$ chasing steps the matrices $Z^{(4)}$ and $H^{(4)}$ are of the form (7.7). The rotation marked by $\times$ is the rotation intuitively popping up as the perturbing one after a left or right removal of a rotation by a similarity transformation

(7.7) $Z^{(4)} =$  and $H^{(4)} =$  .

A closer look at the bottom rotations shows that for both cases, however, the left as well as the right rotation is free. This implies that we can remove either of the bottom rotations. So instead of annihilating the rotation marked in (7.7), one can also remove the ones marked in scheme (7.8):

(7.8) $Z^{(4)} =$  and $H^{(4)} =$  .

After having removed the undesired rotation in $H^{(4)}$ and $Z^{(4)}$ from either (7.7) or (7.8), the patterns as presented in (7.5) are retrieved. The flexibility with the final rotation was already observed in the study of the structure of DA-factorization and also appeared in the explicit version of the DA-algorithm.

Based on the implicit Q-Theorem 5.1 one can prove that the outcome of the implicit DA-version is essentially identical to the outcome of the explicit DA-version. Both the explicit and the implicit methods applied on A result in two unitary similar matrices, admitting a QR-factorization with an identical pattern for factoring their Q-factors. Theorem 5.1 then states that these matrices are essentially identical.

Remark 7.1. Proofs of convergence exist for both ascending as well as descending sequences. Unfortunately, for the moment theoretical results on the convergence for a generic zigzag pattern are not available. Since the chasing involves a swapping between ascending and descending sequences, its convergence seems, however, somehow natural. In the numerical experiments section it will be shown that the combined approach converges neatly and inherits properties from both techniques.

8. Implementation details and numerical examples. Due to the almost unlimited choice of possible positions **p** for a compressed unitary matrix it is impossible to test all algorithms and see how they behave numerically for computing eigenvalues. In the upcoming numerical experiments some selected approaches are used studying the implicit DA-algorithm as presented in this paper.

8.1. Software package. Together with this article a software package is provided for studying the patterns and the flexibility of working with rotational factorizations of unitary matrices. The package provides, for instance, routines for computing the rotational factorization of a unitary matrix, performing a similarity transformation on an arbitrary matrix to obtain a similar QR-factored matrix $\hat{A}$, where Q satisfies a previously defined pattern, computing the DA- and AD-factorization of a compressed unitary matrix, and executing implicit DA-steps on a QR-factored matrix with Q in compressed form. It also contains a tool for visualizing the pattern of a rotational factorization (see Figure 8.1).

In Figure 8.1 patterns for factoring a unitary matrix are shown, visualized with the routine `Gplot`. The left shows the standard pyramid factorization of a unitary matrix $Q \in \mathbb{C}^{20\times 20}$. The right figure shows the CMV-decomposition of a 20×20 pentadiagonal unitary matrix.

The Matlab software can be downloaded from the author's website at `http://people.cs.kuleuven.be/~raf.vandebril/`.

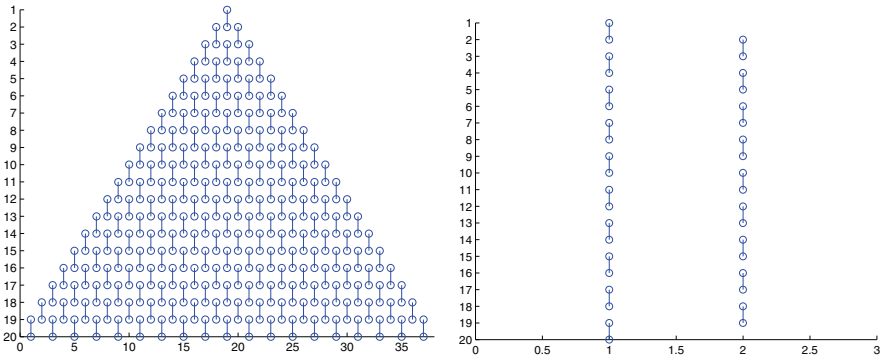


FIG. 8.1. *The left figure is a graphical pyramid shape decomposition of a unitary matrix. The right figure shows the CMV-decomposition of a pentadiagonal unitary matrix.*

8.2. Implementation issues.

8.2.1. Deflation in the *DA*-algorithm. The possibility of subdividing a matrix after a step of the QR-method into two or more problems, which can be treated independently, is referred to as deflation. In the standard Hessenberg case possible deflation is monitored by checking the relative sizes of the subdiagonal elements. Conveying this technique towards the rotations present in the QR-factorization indicates that deflation should be checked by searching for rotations numerically equal to the identity. Indeed, some calculations reveal that a rotation equal to the identity corresponds to a subdivision of the matrix into two submatrices, whose eigenvalues can now be computed independently.

8.2.2. Computational complexity and storage count. Each rotation is stored by two parameters (cosine and the sine), though strictly speaking only a single parameter suffices. For each rotation two additional integers are required for determining the order of appearance in the sequence and the row they act on. Hence, globally storing the QR-factorization of a matrix $A = QR$, with Q in compressed format, uses $1/2n^2 + 9/2n - 6$ parameters.

Important operations in the process are the computation of a Givens rotation, the shift-through operation, and a fusion. Computing a rotation takes $c_r = 6$ operations [17]; a shift-through operation uses 14 operations plus the computation of 2 rotations leading to a cost of $c_s = 14 + 2c_r = 26$ operations. Executing a fusion needs $c_f = 6$ operations.

Complexity count of the unitary similarity transformation. Assume that the QR-factorization of a matrix A is given (roughly $2/3n^3$ operations [17]). Each similarity transformation for annihilation on either the right or the left side uses the same number of operations, so we do not distinguish between those. Assume a single left annihilation step is executed on a matrix of size $\ell \times \ell$. This means that $\ell - 2$ rotations are subjected to removal. To remove the top rotation $\ell - 3$ shift-through operations and a single fusion are performed; annihilation of the next rotation uses $\ell - 4$ shift-through operations and a fusion, and this continues. Transition of a rotation through the upper triangular matrix takes $c_t(\ell) = 6(\ell + 1) + c_r = 6(\ell + 2)$ operations. Globally this gives for a left or right annihilation step the following cost:

$$c_a(\ell) = \sum_{i=1}^{\ell-2} (\ell - 2 - i)c_s + c_f + c_t = 19\ell^2 - 59\ell + 42.$$

Based on the execution of $n-2$ annihilation steps of diminishing sizes a global complexity count is retrieved:

$$c_u = \sum_{\ell=1}^{n-2} c_a(\ell) = \frac{19}{3}n^3 - 58n^2 + \frac{515}{3}n - 162.$$

Taking into consideration the complexity for reducing a matrix to Hessenberg form via rotations (approximately $5n^3$ operations), one can see that the traditional reduction method is about 25% faster than the alternative version based on the rotational QR-factorization.

Complexity count of the DA-method. Consider next an $\ell \times \ell$ matrix on which we want to perform an implicit step of the new DA-method. Again, the left and right annihilation steps are equally expensive, so we will focus only on left annihilations. The initial step, computation of the shift, and the final step are contributing only to the lower order terms in the complexity count so we will neglect them. Each chasing step involves a transferring through the upper triangular matrix, computation of a rotation, and a shift-through operation. So executing $\ell - 2$ chasing steps leads to a complexity of

$$c_s = (\ell - 2)(c_t(\ell) + c_f) = (\ell - 2)(6\ell + 36).$$

The complexity of a traditional chasing step applied on a Hessenberg matrix takes

$$(\ell - 2)(6\ell + 24),$$

which is just few operations cheaper than the new approach.

Deriving the computational complexity of the global DA-procedure is quite cumbersome, since it involves parameters estimating when deflation will take place and so forth. Since all variants are equally expensive, we will therefore in the numerical experiments focus on the number of iterations it takes before deflation can be applied. This is one possible measure for deducing the speed of the approach.

8.3. Numerical experiments for eigenvalue computations. In this section, the accuracy and speed of computing the eigenvalues of some specific matrices are tested. The matrices are first reduced to a compressed QR-factorization after which the DA-algorithm is used to retrieve the eigenvalues.

The software provides a generic DA-algorithm, capable of dealing with any type of pattern of rotations. Only at the end of the chasing should the user specify whether a right or a left annihilation step is taken, with the possibility of changing this during consecutive iterates. Depending on this choice, we distinguish between three types of numerical experiments. In subsection 8.3.1 four fixed patterns are taken for comparison. Subsection 8.3.2 illustrates convergence and accuracy when randomly selecting left or right for the trailing rotation, and in subsection 8.3.3 the pattern will be adaptive, to enhance the convergence speed.

Since at present no adequate convergence theory of this novel method exists, it is impossible to state which left or right selection of the trailing rotation or which preliminary similarity transformation will furnish us with an optimal algorithm. With the numerical experiments presented here we would like, however, to convey some messages on the viability of this generalization of the QR-method, though further research is indispensible. The various experiments illustrate proper convergence of the approach, even for random left-right choice of the final rotation. Comparing several

variants of the DA-algorithm shows us that they are all almost equally accurate, but there is a significant difference in convergence speed. The numerical experiments illustrate also that convergence speed of the DA-method is heavily dependent on the eigenvalue distribution of the original matrix. Moreover, we will also see that with a simple trick, altering the pattern at a specific point, we can speed up convergence.

8.3.1. Accuracy and speed for four particular examples. The almost unlimited number of possibilities forces us to make some specific choices. We have selected four types. For these four variants the same input matrix and identical deflation criteria were used, making a fair comparison possible. We know that the computational complexity is independent of the type of zigzag pattern. Hence, in the following results only the average number of iterations per eigenvalue is plotted. Also the maximum relative accuracy is depicted.

The initial matrix is reduced via a similarity transformation to a compressed QR-factorization after which the DA-algorithm takes over. The four cases are the following:

- *Hessenberg case.* The outcome of the similarity transformation is a Hessenberg matrix. The DA-algorithm does not change the pattern at the end of the sequence of rotations; as a result the sequence remains descending throughout the algorithm. This corresponds to the standard Hessenberg algorithm.
- *Hessenberg-like case.* The matrix is brought via similarity transformation to Hessenberg-like form. The DA-algorithm is run again without changing the pattern at the end of the sequence of rotations. This corresponds to the Hessenberg-like algorithm.
- *CMV-case.* The similarity transformation results in a unitary pentadiagonal matrix. When performing the DA-algorithm the final rotation alters each iterate. In this way one retains the CMV-pattern during the DA-algorithm.
- *Slanted zigzag case.* The compressed QR-factored matrix, resulting from the similarity transformation, is constructed by two left annihilations and a single right annihilation. The corresponding DA-algorithm alters the direction every two steps, so after two DA-steps the direction of the trailing rotation changes.

Deflation is applied when the absolute value of the sine of a rotation is below 10^{-14}, this criterion is identical for all test cases. The algorithm only deflates blocks of sizes at most 2. As shift the Rayleigh shift is taken. The DA-method is identical for all four versions, only the left-right position of the final rotation differs.

In the first numerical experiment a random symmetric matrix is generated with eigenvalues equally distributed in the interval $[0, 1]$. The problem sizes vary from 20 to 500. The four variants are tested on the same matrix. Figure 8.2 shows the accuracy and the average number of iterations. Taking into consideration that the DA-steps are equally expensive for each method, the average number of iterations is a good measure for deducing the speed of the corresponding approach.

All four methods provide almost equally accurate results. Only the number of iterations differs significantly. An inappropriate conclusion would state that the Hessenberg-like case is the best, since the DA-algorithm is related to a QR-method and the convergence of QR-methods is heavily dependent on the ratios between eigenvalues. Because the convergence of the Hessenberg-like approach is related to the inverse of the matrix A, it seems natural that it outperforms the other methods in the case of equal spaced eigenvalues. To illustrate this, a second experiment is performed, with eigenvalues the inverses of the eigenvalues of the first experiment. Figure 8.3 depicts the results. As expected the Hessenberg case outperforms now the Hessenberg-

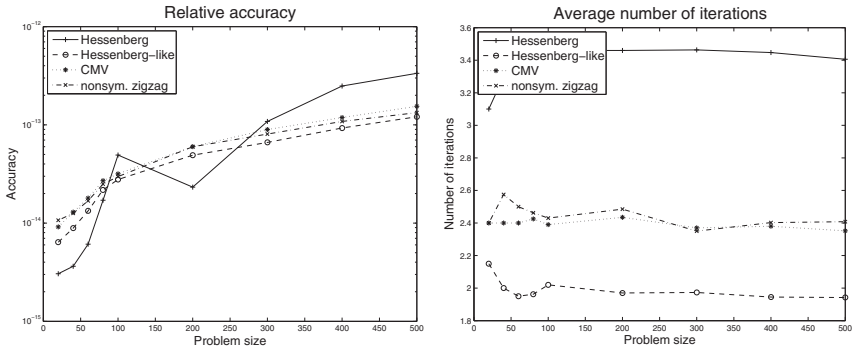


FIG. 8.2. *Symmetric matrices with equal spaced eigenvalues in the interval* $[0, 1]$.

like case. The two other approaches are somewhere in the middle of the figure as their convergence relies on powers of both A and A^{-1}. Comparing the average numbers of iterations in Figures 8.2 and 8.3, we see that the extremes differ quite a lot. The averages range from 3.4 to 2. In practice, this is a nontrivial reduction in computational time. As already mentioned, the convergence speed is dependent, however, heavily on the eigenvalue distribution. Further research is required to fully understand the interaction between the eigenvalue distribution and the rotational patterns chosen. In this fashion, one might possibly be able to tune the DA-algorithm to get the average number of iterations as low as possible, gaining thereby notable computing time.

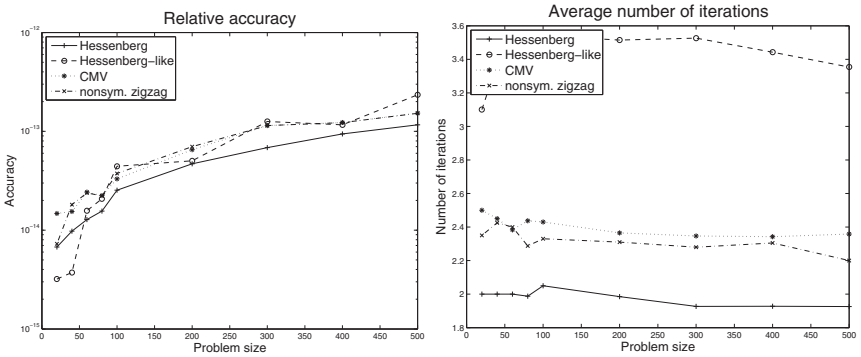


FIG. 8.3. *Symmetric matrices with eigenvalues the inverses of equal spaced numbers in* $[0, 1]$.

8.3.2. Random left-right selection of the final rotation. In the previous numerical experiments, some well-determined patterns regulated the iterative steps. In the next experiment the position of the final rotation in each QR-step is chosen at random; that is, decisions to go right or left are taken randomly. With this example we would like to illustrate that periodicity of the patterns is not required for convergence; also for arbitrarily chosen formats altering throughout the process we will have convergence. The eigenvalue distributions of the previous examples are repeated. Figure 8.4 displays the relative accuracy and the average number of iterations for equal spaced eigenvalues in $[0, 1]$. As benchmarks the Hessenberg and Hessenberg-like case are depicted again together with three independent random runs. Since there is no actual connection between each random test case and to avoid cluttering of the figures

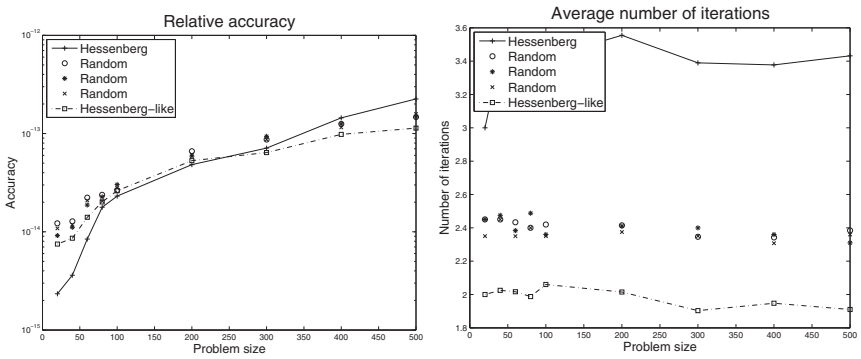


FIG. 8.4. *Behavior of randomly selected patterns determining the QR-steps on equal spaced eigenvalues in* $[0, 1]$.

no lines connecting the individual random samples are drawn. To provide, however, detailed information on each random trial, extra tables are included. For each instance the problem size, the average number of iterations, the number of times left or right, was chosen and the number of direction changes encountered in the left-right choices during the execution of the algorithm is depicted in Table 8.1. Figure 8.5 and Table 8.2 display similar information, but now for eigenvalues as the inverses of equal spaced numbers in $[0, 1]$.

TABLE 8.1
Detailed information on the random tests in the Figures 8.4.

Marker	Problem Size	Avg. number of iterations	Number of left choices	Number of right choices	Number of Direction Changes
○	20	2.45	20	29	23
○	40	2.45	45	53	59
○	60	2.43	67	79	78
○	80	2.40	102	90	98
○	100	2.42	127	115	129
○	200	2.42	221	262	268
○	300	2.35	334	370	335
○	400	2.34	460	477	469
○	500	2.38	601	591	623
⋆	20	2.45	26	23	30
⋆	40	2.48	47	52	43
⋆	60	2.38	74	69	72
⋆	80	2.49	98	101	105
⋆	100	2.36	120	116	110
⋆	200	2.41	236	246	237
⋆	300	2.40	347	373	350
⋆	400	2.36	430	514	469
⋆	500	2.31	600	555	583
×	20	2.35	26	21	27
×	40	2.45	59	39	42
×	60	2.35	77	64	80
×	80	2.40	99	93	94
×	100	2.35	117	118	104
×	200	2.38	233	242	232
×	300	2.35	358	347	330
×	400	2.31	462	461	476
×	500	2.36	574	606	578

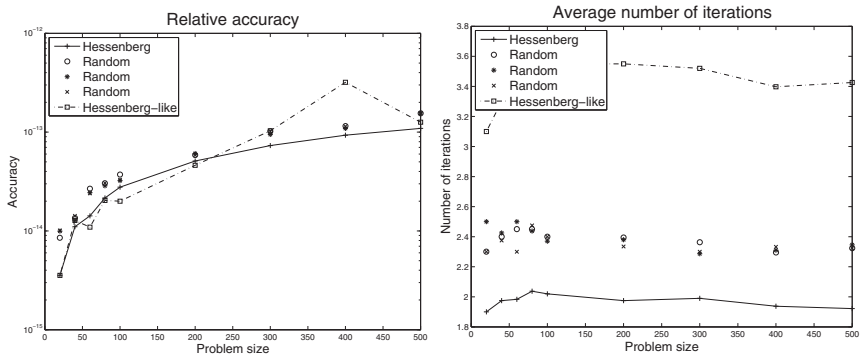


FIG. 8.5. *Behavior of randomly selected patterns determining the QR-steps with eigenvalues the inverses of equal spaced numbers in* $[0, 1]$.

TABLE 8.2
Detailed information on the random tests in the Figures 8.5.

Marker	Problem Size	Avg. number of iterations	Number of left choices	Number of right choices	Number of Direction Changes
○	20	2.30	25	21	17
○	40	2.40	46	50	46
○	60	2.45	75	72	60
○	80	2.45	87	109	90
○	100	2.40	134	106	110
○	200	2.40	239	240	249
○	300	2.36	349	360	343
○	400	2.29	445	473	434
○	500	2.32	597	565	575
⋆	20	2.50	30	20	24
⋆	40	2.42	51	46	53
⋆	60	2.50	73	77	77
⋆	80	2.44	99	96	99
⋆	100	2.37	114	123	106
⋆	200	2.38	217	259	230
⋆	300	2.29	343	343	356
⋆	400	2.31	475	449	456
⋆	500	2.35	596	577	597
×	20	2.30	24	22	23
×	40	2.38	51	44	41
×	60	2.30	69	69	62
×	80	2.48	96	102	105
×	100	2.40	110	130	107
×	200	2.33	234	233	234
×	300	2.30	336	354	361
×	400	2.33	466	467	459
×	500	2.32	580	582	598

The Figures 8.4 and 8.5 show that the convergence of the random case is in between the Hessenberg and Hessenberg-like case. A closer look at Tables 8.1 and 8.2 reveals that the more a random process approaches (depending on the number of left and right choices) the Hessenberg or Hessenberg-like case, the more it resembles its convergence.

8.3.3. Flexible adaptation of the pattern. In the final numerical experiment we changed the DA-algorithm and made it adaptive. Instead of selecting the position

of the final rotation we alter it depending on the average number of iterations. Starting, for example, with a Hessenberg matrix, we retain the structure and do not alter the direction of the trailing rotation. It is assumed that on average 2.5 DA-steps are needed to find all eigenvalues. After 10% of this number, $n/4$ DA-steps, we check the average number of iterations before convergence occurs. If this is too high we alter the direction. In the Hessenberg case it means that we change the direction of the final rotation, hoping that by performing this action we can still reduce the average number of iterations.

This method was tested on a single example with equal spaced eigenvalues. To compare the flexible approach the average number of iterations for the Hessenberg and Hessenberg-like approach is also shown (Figure 8.6). Clearly, changing the direction has a significant impact on the average number of iterations and thus also on the global computational time.

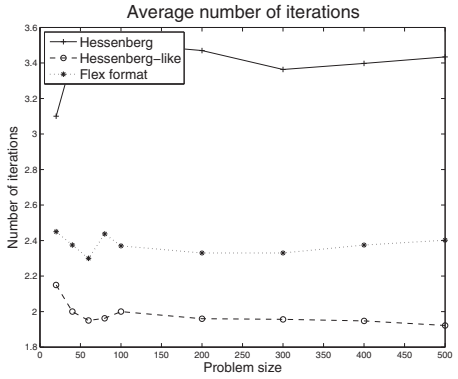


FIG. 8.6. *Convergence determined by a flexible pattern on equal spaced eigenvalues in* $[0, 1]$.

9. Conclusions and future research. In this article a new procedure, based on the QR-factorization of a given matrix, is presented for transforming a matrix via unitary similarity transformations to a compressed format. This similarity transformation is a generalization of the reductions to Hessenberg and Hessenberg-like form, and it can achieve, for instance, the CMV-pattern in the decomposition of the unitary matrix. Uniqueness of the reduction is proved. Based on the factorization of the unitary matrix a new procedure, generalizing the QR-algorithm, is given. An implicit version of this method is presented. Numerical experiments show the viability of this approach revealing the impact of the eigenvalue distribution on the convergence of the DA-algorithm.

The numerics clearly illustrate the significant impact of the pattern choice on the convergence speed. Intelligently choosing the pattern can result in a dramatic decrease in the number of iterations. Unfortunately, the research associated with this algorithm is not yet finished and we are currently unable to effectively construct an autonomous algorithm choosing the pattern on the fly to obtain the fastest convergence possible. To tackle this problem we need to address important open theoretical questions such as a proof of convergence (illustrated by the numerics), an analysis of the convergence speed, a theoretical foundation on which side to choose the trailing rotation, and how to pick the shifts. All these questions will be the subject of further research.

REFERENCES

[1] G. S. Ammar, W. B. Gragg, and L. Reichel, *On the eigenproblem for orthogonal matrices*, in Proceedings of the 25th IEEE Conference on Decision & Control, New York, 1986, pp. 1963–1966.

[2] G. S. Ammar, W. B. Gragg, and L. Reichel, *Constructing a unitary Hessenberg matrix from spectral data*, in Numerical Linear Algebra, Digital Signal Processing and Parallel Algorithms, Comput. Syst. Sci. 70, G. H. Golub and P. Van Dooren, eds., Springer-Verlag, Berlin, Germany, 1991, pp. 385–395.

[3] G. S. Ammar and V. Mehrmann, *A geometric perspective on condensed forms for Hamiltonian matrices*, in Computation and Control II: Proceedings of the Second Bozeman Conference, K. L. Bowers and J. Lund, eds., Birkhäuser, Basel, 1991, pp. 1–11.

[4] R. C. Barroso and S. Delvaux, *Orthogonal Laurent polynomials on the unit circle and snake-shaped matrix factorizations*, J. Approx. Theory, 161 (2009), pp. 65–87.

[5] D. A. Bini, Y. Eidelman, L. Gemignani, and I. C. Gohberg, *Fast QR eigenvalue algorithms for Hessenberg matrices which are rank-one perturbations of unitary matrices*, SIAM J. Matrix Anal. Appl., 29 (2007), pp. 566–585.

[6] K. Braman, R. Byers, and R. Mathias, *The multishift QR algorithm. Part* I*: Maintaining well-focused shifts and level 3 performance*, SIAM J. Matrix Anal. Appl., 23 (2002), pp. 929–947.

[7] K. Braman, R. Byers, and R. Mathias, *The multishift QR algorithm. Part* II*: Aggressive early deflation*, SIAM J. Matrix Anal. Appl., 23 (2002), pp. 948–973.

[8] A. Bunse-Gerstner and L. Elsner, *Schur parameter pencils for the solution of the unitary eigenproblem*, Linear Algebra Appl., 154–156 (1991), pp. 741–778.

[9] M. J. Cantero, L. Moral, and L. Velazquez, *Five-diagonal matrices and zeros of orthogonal polynomials on the unit circle*, Linear Algebra Appl., 362 (2003), pp. 29–56.

[10] S. Delvaux and M. Van Barel, *The explicit QR-algorithm for rank structured matrices*, Technical report TW459, Department of Computer Science, Katholieke Universiteit Leuven, Celestijnenlaan 200A, 3000 Leuven (Heverlee), Belgium, 2006.

[11] S. Delvaux and M. Van Barel, *Rank structures preserved by the QR-algorithm: The singular case*, J. Comput. Appl. Math., 189 (2006), pp. 157–178.

[12] S. Delvaux and M. Van Barel, *Structures preserved by the QR-algorithm*, J. Comput. Appl. Math., 187 (2006), pp. 29–40.

[13] J. W. Demmel, *Applied Numerical Linear Algebra*, SIAM, Philadelphia, PA, 1997.

[14] Y. Eidelman, L. Gemignani, and I. C. Gohberg, *On the fast reduction of a quasiseparable matrix to Hessenberg and tridiagonal forms*, Linear Algebra Appl., 420 (2007), pp. 86–101.

[15] Y. Eidelman, I. C. Gohberg, and V. Olshevsky, *The QR iteration method for Hermitian quasiseparable matrices of an arbitrary order*, Linear Algebra Appl., 404 (2005), pp. 305–324.

[16] H. Fassbender, *Symplectic Methods for the Symplectic Eigenvalue Problem*, Kluwer Academic Publishers, Norwell, MA, 2002.

[17] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd ed., Johns Hopkins University Press, Baltimore, MD, 1996.

[18] N. Mastronardi, S. Chandrasekaran, and S. Van Huffel, *Fast and stable two-way algorithm for diagonal plus semi-separable systems of linear equations*, Numer. Linear Algebra Appl., 8 (2001), pp. 7–12.

[19] B. N. Parlett, *The Symmetric Eigenvalue Problem*, Classics Appl. Math. 20, SIAM, Philadelphia, PA, 1998.

[20] B. Simon, *CMV matrices: Five years after*, J. Comput. Appl. Math., 208 (2007), pp. 120–154.

[21] G. W. Stewart, *Matrix Algorithms, Volume II: Eigensystems*, SIAM, Philadelphia, PA, 2001.

[22] M. Van Barel, R. Vandebril, and N. Mastronardi, *An orthogonal similarity reduction of a matrix into semiseparable form*, SIAM J. Matrix Anal. Appl., 27 (2005), pp. 176–197.

[23] M. Van Barel, R. Vandebril, P. Van Dooren, and K. Frederix, *Implicit double shift QR-algorithm for companion matrices*, Numer. Math., 116 (2010), pp. 177–212.

[24] R. Vandebril, *The interplay of Givens rotations and QR-factorizations*, Technical report, Department of Computer Science, Katholieke Universiteit Leuven, Celestijnenlaan 200A, 3001 Leuven (Heverlee), Belgium, 2010.

[25] R. Vandebril and G. M. Del Corso, *An implicit multishift QR-algorithm for Hermitian plus low rank matrices*, SIAM J. Sci. Comput., 32 (2010), pp. 2190–2212.

[26] R. Vandebril, M. Van Barel, and N. Mastronardi, *An implicit QR-algorithm for symmetric semiseparable matrices*, Numer. Linear Algebra Appl., 12 (2005), pp. 625–658.

[27] R. Vandebril, M. Van Barel, and N. Mastronardi, *Matrix Computations and Semiseparable Matrices, Volume I: Linear Systems*, Johns Hopkins University Press, Baltimore, MD, 2008.
[28] R. Vandebril, M. Van Barel, and N. Mastronardi, *Matrix Computations and Semiseparable Matrices, Volume II: Eigenvalue and Singular Value Methods*, Johns Hopkins University Press, Baltimore, MD, 2008.
[29] R. Vandebril, M. Van Barel, and N. Mastronardi, *A parallel QR-factorization/solver of structured rank matrices*, Electron. Trans. Numer. Anal., 30 (2008), pp. 144–167.
[30] R. Vandebril, M. Van Barel, and N. Mastronardi, *A rational QR-iteration without inversion*, Numer. Math., 110 (2008), pp. 561–575.
[31] R. Vandebril, M. Van Barel, and N. Mastronardi, *A new iteration for computing the eigenvalues of semiseparable (plus diagonal) matrices*, Electron. Trans. Numer. Anal., 33 (2009), pp. 126–150.
[32] D. S. Watkins, *Understanding the QR algorithm*, SIAM Rev., 24 (1982), pp. 427–440.
[33] D. S. Watkins, *Some perspectives on the eigenvalue problem*, SIAM Rev., 35 (1993), pp. 430–471.
[34] D. S. Watkins, *Bulge exchanges in algorithms of QR type*, SIAM J. Matrix Anal. Appl., 19 (1998), pp. 1074–1096.
[35] D. S. Watkins, *On the reduction of a Hamiltonian matrix to Hamiltonian Schur form*, Electron. Trans. Numer. Anal., 23 (2006), pp. 141–157.
[36] D. S. Watkins, *The Matrix Eigenvalue Problem: GR and Krylov Subspace Methods*, SIAM, Philadelphia, PA, 2007.
[37] D. S. Watkins, *The QR algorithm revisited*, SIAM Rev., 50 (2008), pp. 133–145.
[38] D. S. Watkins and L. Elsner, *Chasing algorithms for the eigenvalue problem*, SIAM J. Matrix Anal. Appl., 12 (1991), pp. 374–384.
[39] D. S. Watkins and L. Elsner, *Convergence of algorithms of decomposition type for the eigenvalue problem*, Linear Algebra Appl., 143 (1991), pp. 19–47.
[40] D. S. Watkins and L. Elsner, *Theory of decomposition and bulge-chasing algorithms for the generalized eigenvalue problem*, SIAM J. Matrix Anal. Appl., 15 (1994), pp. 943–967.
[41] J. H. Wilkinson, *The Algebraic Eigenvalue Problem*, Oxford University Press, New York, 1988.